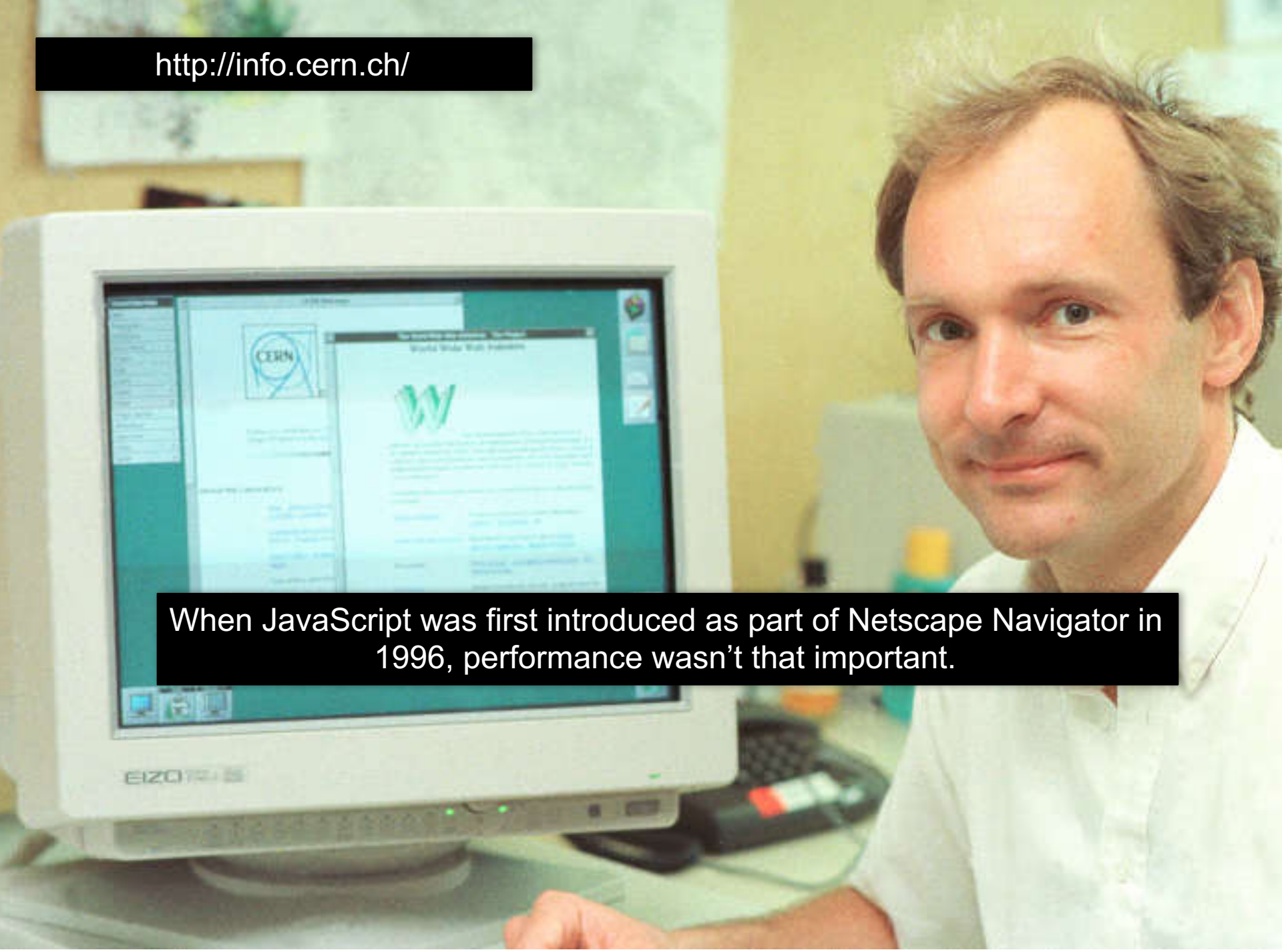
NEW MEDIA &
COMMUNICATION
TECHNOLOGY

Front End Web Development

NEW MEDIA &
COMMUNICATION
TECHNOLOGY

Performantie


http://info.cern.ch/
When JavaScript was first introduced as part of Netscape Navigator in 1996, performance wasn’t that important.

Back-end

5%

Front-end

95%

Metrics


Split/Reset
Start/Stop
OREGON SCIENTIFIC
HR MIN SEC
LAP
0 10
0:00' 16.36"
0:01'21"38
SPLT
04
0:01'21"38
SEGMENT

Metrics

Hou metrics bij wanneer je web applicatie traag is of wordt.

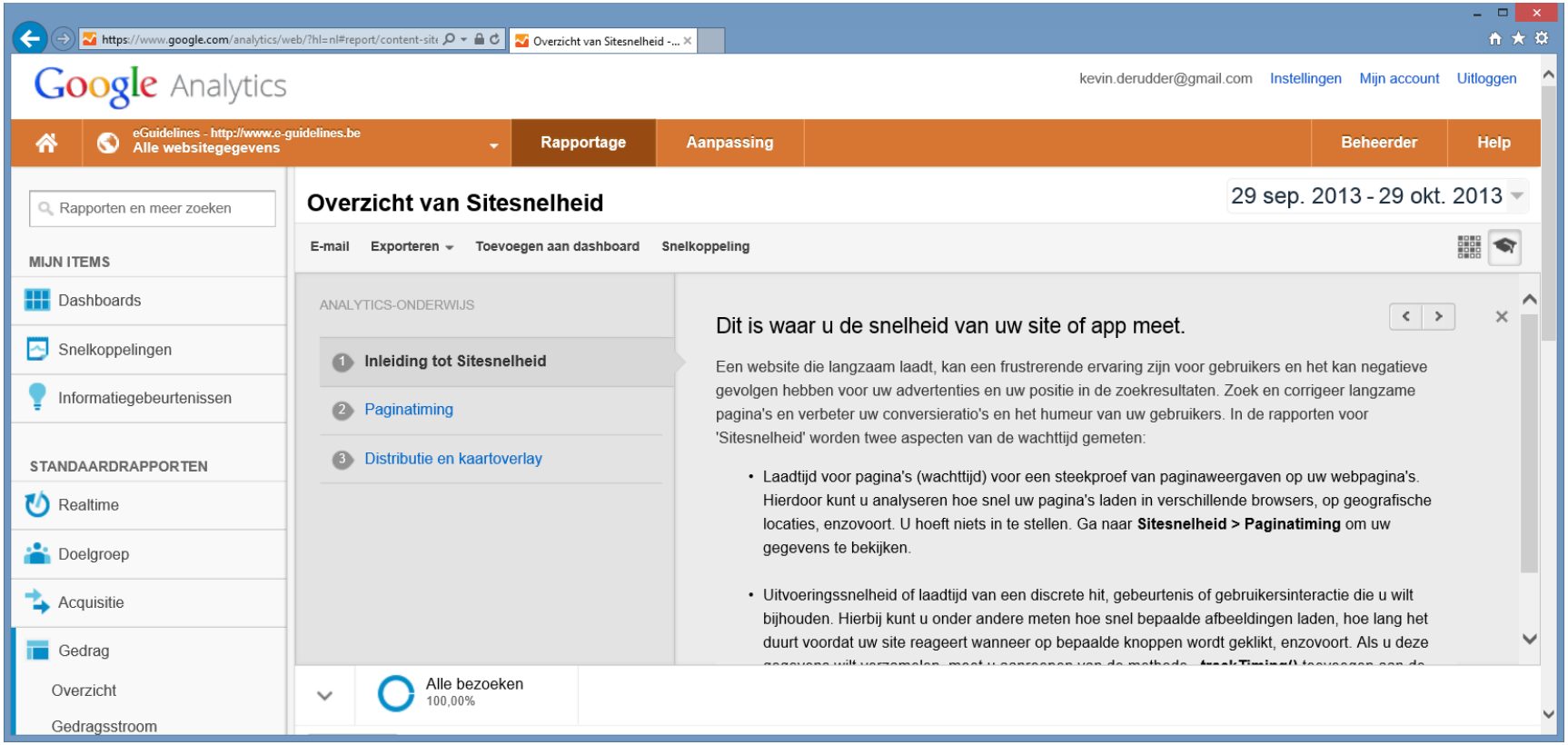
meten is weten

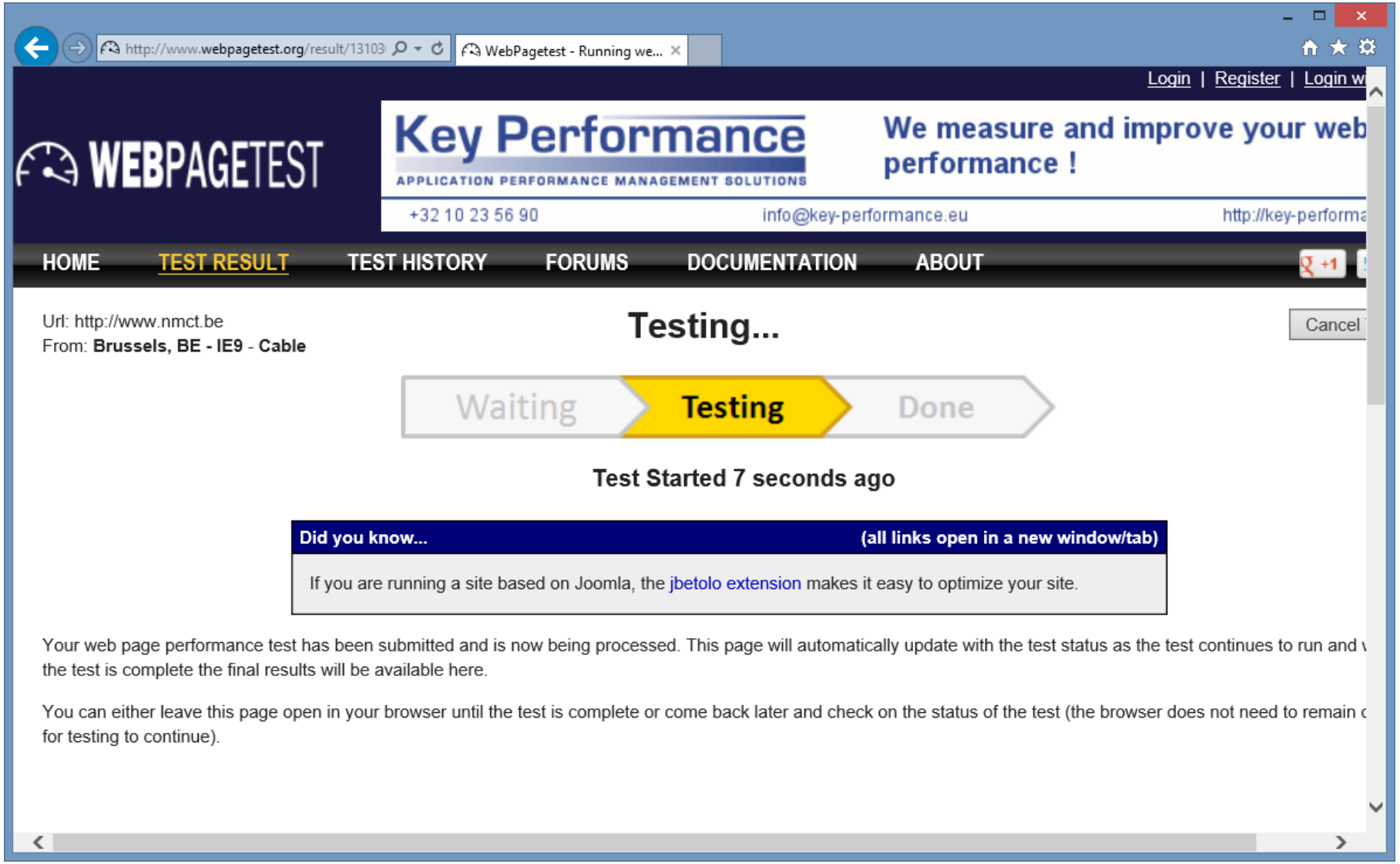
Google analytics

F12 tools

webpagetest

...


https://www.google.com/analytics/web/?hl=nl#report/content-site
Overzicht van Sitesnelheid -...
Google Analytics
kevin.derudder@gmail.com
Instellingen
Mijn account
Uitloggen
eGuidelines - http://www.e-guidelines.be
Alle websitegegevens
Rapportage
Aanpassing
Beheerder
Help
Rapporten en meer zoeken
MIJN ITEMS
Dashboards
Snelkoppelingen
Informatiegebeurtenissen
STANDAARDRAPPORTEN
Realtime
Doelgroep
Acquisitie
Gedrag
Overzicht
Gedragsstroom
Overzicht van Sitesnelheid
29 sep. 2013 - 29 okt. 2013
E-mail
Exporteren
Toevoegen aan dashboard
Snelkoppeling
ANALYTICS-ONDERWIJS
1 Inleiding tot Sitesnelheid
2 Paginatiming
3 Distributie en kaartoverlay
Dit is waar u de snelheid van uw site of app meet.
Een website die langzaam laadt, kan een frustrerende ervaring zijn voor gebruikers en het kan negatieve gevolgen hebben voor uw advertenties en uw positie in de zoekresultaten. Zoek en corrigeer langzame pagina's en verbeter uw conversieratio's en het humeur van uw gebruikers. In de rapporten voor 'Sitesnelheid' worden twee aspecten van de wachttijd gemeten:
• Laadtijd voor pagina's (wachttijd) voor een steekproef van paginaweergaven op uw webpagina's. Hierdoor kunt u analyseren hoe snel uw pagina's laden in verschillende browsers, op geografische locaties, enzovoort. U hoeft niets in te stellen. Ga naar Sitesnelheid > Paginatiming om uw gegevens te bekijken.
• Uitvoeringssnelheid of laadtijd van een discrete hit, gebeurtenis of gebruikersinteractie die u wilt bijhouden. Hierbij kunt u onder andere meten hoe snel bepaalde afbeeldingen laden, hoe lang het duurt voordat uw site reageert wanneer op bepaalde knoppen wordt geklikt, enzovoort. Als u deze
Alle bezoeken
100,00%


http://www.webpagetest.org/result/13103
WebPagetest - Running we...
Login | Register | Login wi
WEBPAGETEST
Key Performance
APPLICATION PERFORMANCE MANAGEMENT SOLUTIONS
We measure and improve your web performance !
+32 10 23 56 90
info@key-performance.eu
http://key-performa
HOME
TEST RESULT
TEST HISTORY
FORUMS
DOCUMENTATION
ABOUT
+1
Url: http://www.nmct.be
From: Brussels, BE - IE9 - Cable
Testing...
Cancel
Waiting
Testing
Done
Test Started 7 seconds ago
Did you know...
(all links open in a new window/tab)
If you are running a site based on Joomla, the jbetolo extension makes it easy to optimize your site.
Your web page performance test has been submitted and is now being processed. This page will automatically update with the test status as the test continues to run and v
the test is complete the final results will be available here.
You can either leave this page open in your browser until the test is complete or come back later and check on the status of the test (the browser does not need to remain c
for testing to continue).

VERPLICHTE LECTUUR!!!

https://developers.google.com/speed/pagespeed/

http://dev.opera.com/articles/view/efficient-javascript/

LOOPS

Hoeveel verschillende loops ken je in JavaScript?

Welke?

```
for (var i = 0 ; i < 10 ; i++) {

}
```

```
var i = 0;
while (i < 10) {
    // body
    i++;
}
```

```
var i = 0;
do {
    //body
} while (i++ < 10);
```

```
for (var prop in object) {
    //body
}
```

Welke loop is de traagste en welke is de snelste?

De for-in is de traagste en kan perfect vervangen worden door een andere loop

Zorg ervoor dat zo weinig mogelijk zaken moeten gebeuren tijdens de loop!

```
for (var i = 0 ; i < items.length ; i++)
{
    doeIets(items[i]);
}
```

1. Een property lookup (items.length)
2. Een vergelijking (i < items.length)
3. Een vergelijking(i < items.length == true)
4. Increment (i++)
5. Array lookup (items[i])
6. Functie oproepen(doeIets(items[i]))

herhaald per iteratie

Beperk property lookups tot het minimum

```
for (var i = 0, len = items.length ; i < len ; i++)
{
    doeIets(items[i]);
}
```

Afhankelijk van de lengte van de array kan dit 25% tijdswinst opleveren

We gebruikten de for als voorbeeld,
maar dit geldt ook voor de while en do while

Decrement ipv Increment

Vergelijking met 0

```
for (var i = items.length ; i > 0 ;i-- )
{
    doeIets(items[i]);
}
```

```
var i = items.length;
while (i--) {
    // body
    doeIets(items[i]);
}
```

Afhankelijk van de lengte van de array kan dit 50-60% tijdswinst opleveren

Decrement ipv Increment

Elke vergelijking is nu een simpele vergelijking met 0. Een vergelijking moet de waarde true opleveren en elke keer de waarde van i verschillend is van 0, levert dit een true op.

Dit zorgt ervoor dat de waarde 0 automatisch false is.

Bij increment heb je dus twee stappen:

1. i < lengte van de array
2. Is dit gelijk aan true

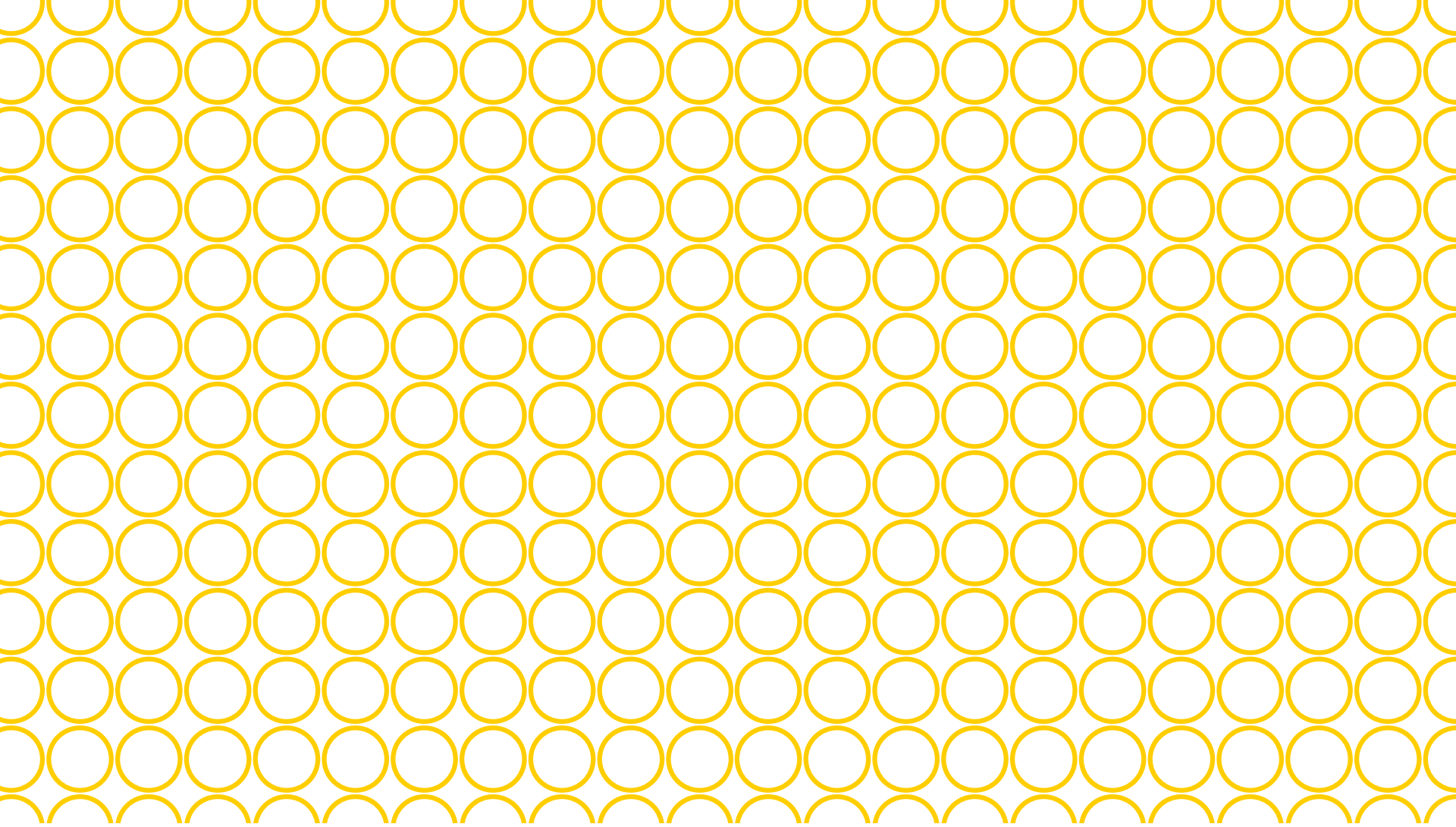
Bij decrement heb je maar 1 stap niet meer, nl: is de waarde true

Duff’s device

Beperk het aantal iteraties

Duff's device

```
//credit: Jeff Greenberg
var iterations = Math.floor(items.length / 8),
startAt = items.length % 8,
i = 0;
do {
    switch (startAt) {
        case 0: doeIets(items[i++]);
        case 7: doeIets(items[i++]);
        case 6: doeIets(items[i++]);
        case 5: doeIets(items[i++]);
        case 4: doeIets(items[i++]);
        case 3: doeIets(items[i++]);
        case 2: doeIets(items[i++]);
        case 1: doeIets(items[i++]);
    }
    startAt = 0;
} while (--iterations);
```



CONDITIONALS

If-else vs switch

De switch is sneller bij een hoger aantal condities

Elke conditie die je toevoegt is zwaarder voor een if-else dan voor een switch

Wanneer je meer als 2 condities hebt, is een switch interessant om te gebruiken

Is trouwens ook veel leesbaarder

If-else optimalizatie

1. Beperk het aantal condities

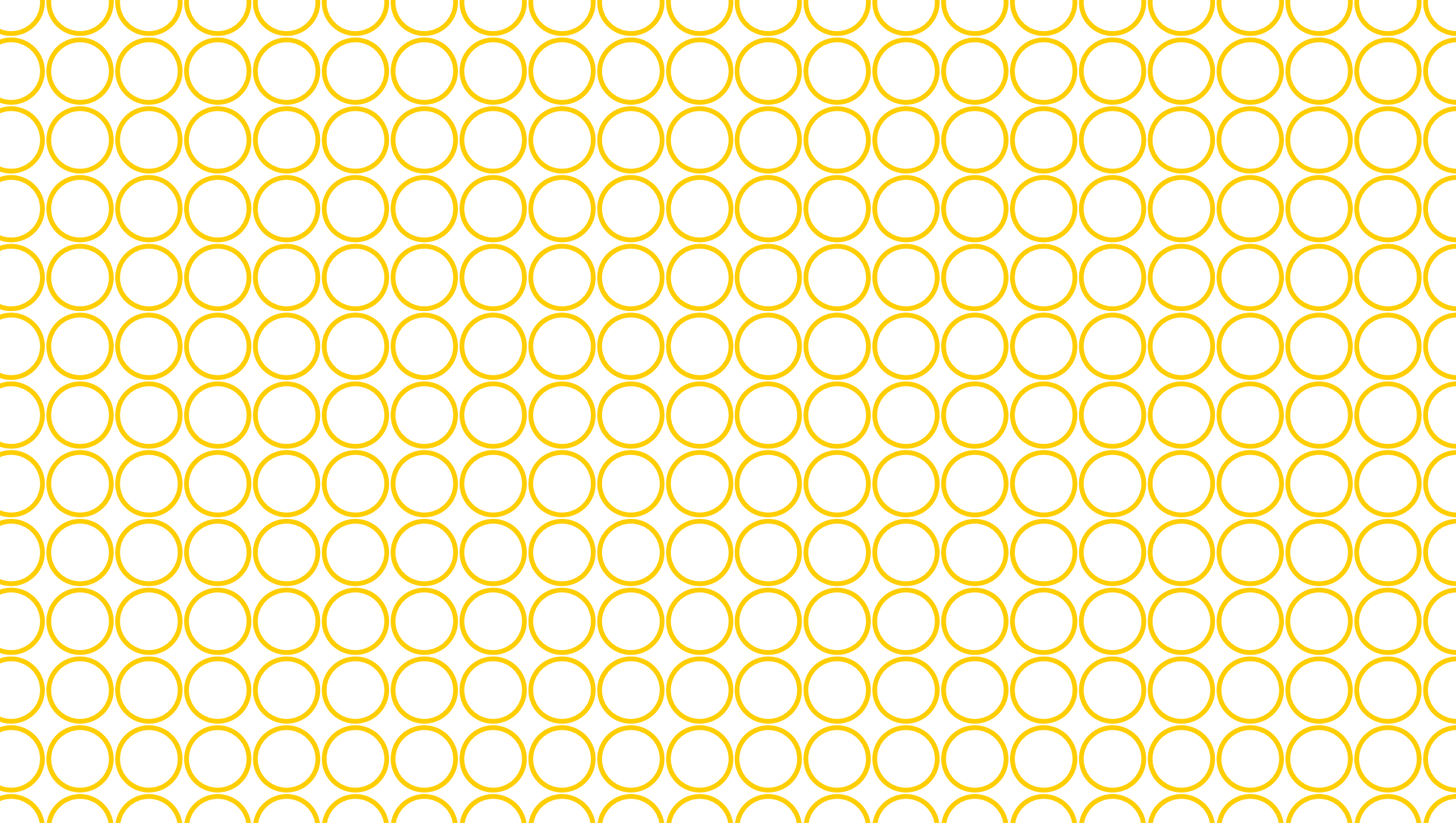
2. Zorg dat de meest voor de hand liggende conditie bovenaan staat

3. Beter meerdere if elsen genest dan onder elkaar

```
if (value == 0){
    return result0;
} else if (value == 1){
    return result1;
} else if (value == 2){
    return result2;
} else if (value == 3){
    return result3;
} else if (value == 4){
    return result4;
} else if (value == 5){
    return result5;
} else if (value == 6){
```

VS

```
if (value < 6){
    if (value < 3){
        if (value == 0){
            return result0;
        } else if (value == 1){
            return result1;
        } else {
            return result2;
        }
    } else {
        if (value == 3){
            return result3;
        } else if (value == 4){
            return result4;
        } else {
            return result5;
        }
    }
} else {
    if (value < 8){
        if (value == 6){
            return result6;
        } else {
            return result7;
        }
    } else {
        if (value == 8){
            return result8;
        } else if (value == 9){
            return result9;
        } else {
            return result10;
```



STRINGS

Methode	Voorbeeld
+	s = ‘a’ + ‘b’ + ‘c’;
+=	s = ‘a’; s+=‘b’; s+=‘c’;
array.join	s = [‘a’, ‘b’, ‘c’].join(‘’);
string.concat	s = "a"; s= s("b", "c");

Wat gebeurt er bij +=

```
s += "a" + "b";
```

1. Tijdelijke string wordt in het geheugen gemaakt
2. De waarde “ab” wordt aan die string toegewezen
3. De tijdelijke string wordt aan s geconcateneerd
4. Het resultaat wordt aan s toegekend

Snellere oplossing

```
s += "a";
s += "b";
```

10 – 40% sneller in meeste browsers doordat geen tijdelijke string moet aangemaakt worden

Page Reflows

Page Reflow

Reflow en Repaint is het process die nodig is om uw HTML pagina te renderen

Probeer dit zo veel mogelijk te beperken!!!

Volgens Opera: 1 van de grote redenen trage websites

http://www.youtube.com/watch?v=nJtBUHyNBxs

http://www.youtube.com/watch?v=ZTnIxIA5KGw

http://www.youtube.com/watch?v=dndeRnzkJDU

Wanneer

Een **repaint** gebeurt wanneer er de visibility van een element gewijzigd wordt, zonder de layout te wijzigen

outline, visibility of background color

De browser moet hierdoor de gehele DOM aflopen

Wanneer

Een **repaint** is nog erger omdat er ook een wijziging in de layout is

Zowel **JavaScript** als **CSS** kunnen reflows of repaints veroorzaken

Wat veroorzaakt een reflow

Resizing window

Changing font

Toevoegen of verwijderen van een stylesheet

Content wijzigingen, zelf wanneer de user iets intypt

CSS pseudo classes zoals bijvoorbeeld hover

Class attribute wijzigen

DOM manipulatie

offsetWidth en offsetHeight berekenen

Property van het stijl attribuut wijzigen

Mogelijke oplossingen

GOED

```
var content = "";
for (var i = 0 ; i < 1000 ; i++) {
    content += "product" + i;
}
$("#theElement").append(content);
```

SLECHT

```
for (var i = 0 ; i < 1000 ; i++) {
    $("#theElement").append("product" + i);
}
```

Mogelijke oplossingen

GOED

```
function changeStyle(element) {
    $(element).addClass("selectedAnchor");
}
```

SLECHT

```
function changeStyle(element) {
    element.style.fontWeight = "bold";
    element.style.textDecoration = "none";
    element.style.color = "#000";
}
```

Mogelijke oplossingen

GOED

```
function addElement(parentElement, liText, liClass) {
    var li = document.createElement("li");
    li.className = anchorClass;
    parentElement.appendChild(li);
}
```

SLECHT

```
function addElement(parentElement, liText, liClass) {
    var li = document.createElement("li");
    parentElement.appendChild(li);
    li.className = anchorClass;
}
```

Loading Scripts

Laden van

Tot nu toe hebben we gezegd dat je beter de scripts referenties onderaan de pagina plaatst!

Dit omdat wanneer de parser een external script tegenkomt hij verdere parsing stopt.

script moet gedownload worden

script moet uitgevoerd worden

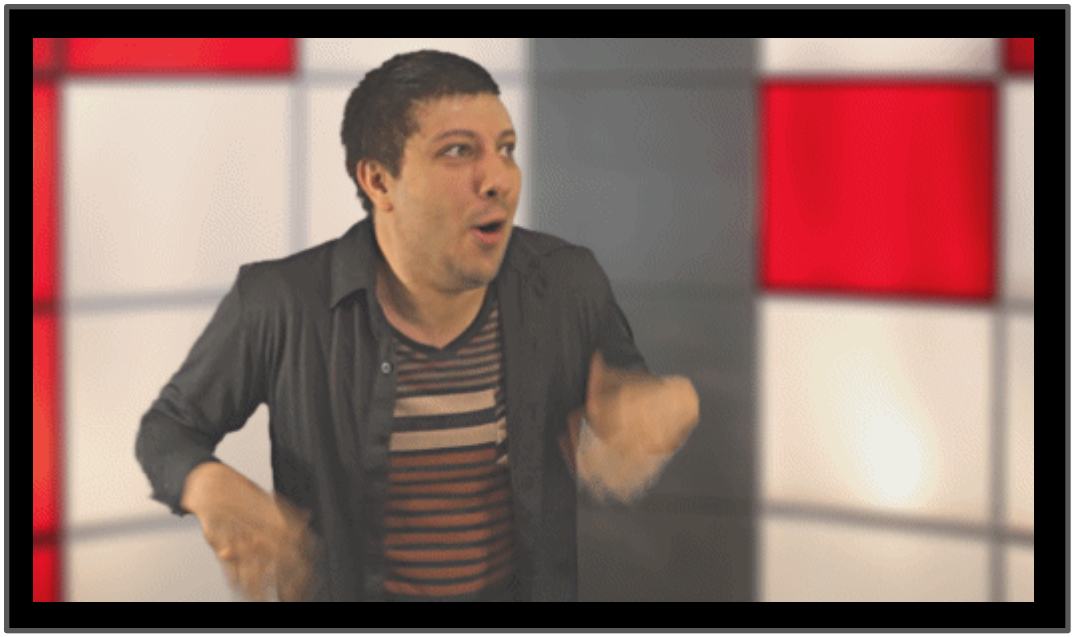
https://www.webkit.org/blog/1395/running-scripts-in-webkit/


Geen nood, er zijn oplossingen

JavaScript achteraf downloaden

via AJAX

Async of Defer



Defer en Async JavaScript

```
<script src="demo.js" defer ></script>

<script src="demo.js" async ></script>
```

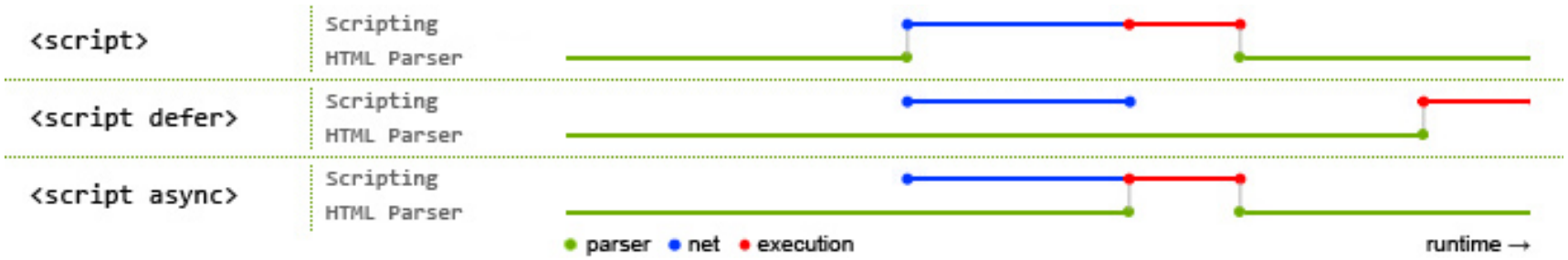
Defer JavaScript

Description

Warning: Defering JavaScript can often dramatically improve the rendering speed of a site, but careful evaluation is required to ensure the site continues to operate properly. The limitations are described below.

Defer JavaScript tries to defer JavaScript execution until page load. It defers this by changing the `type` and `src` attributes of `<script>` elements on the HTML page to `pagespeed_orig_type` and `pagespeed_orig_src` respectively. It also adds a new `type` attribute whose value is set to `text/psajs`. A `window.onload` handler is added to the HTML, which executes all the deferred scripts.

Defer JavaScript doesn't defer a `script` tag if it has the `pagespeed_no_defer` attribute. This is useful when a `script` tag needs to be executed while loading the page. For example, a `script` tag may be updating the main content dynamically as a slideshow of images in the visible content of the page. Usage:



Normaal

Parser stop met het renderen van HTML tot script gedownload en uitgevoerd is

Defered

Script wordt pas uitgevoerd wanneer de HTML parser klaar is.
Voordeel is dat de DOM beschikbaar is

Async

HTML parsing stopt enkel voor het uitvoeren van de JavaScript en gaat dan verder.
Ideaal voor google analytics

Gevaar

Het gevaar is zeer reëel

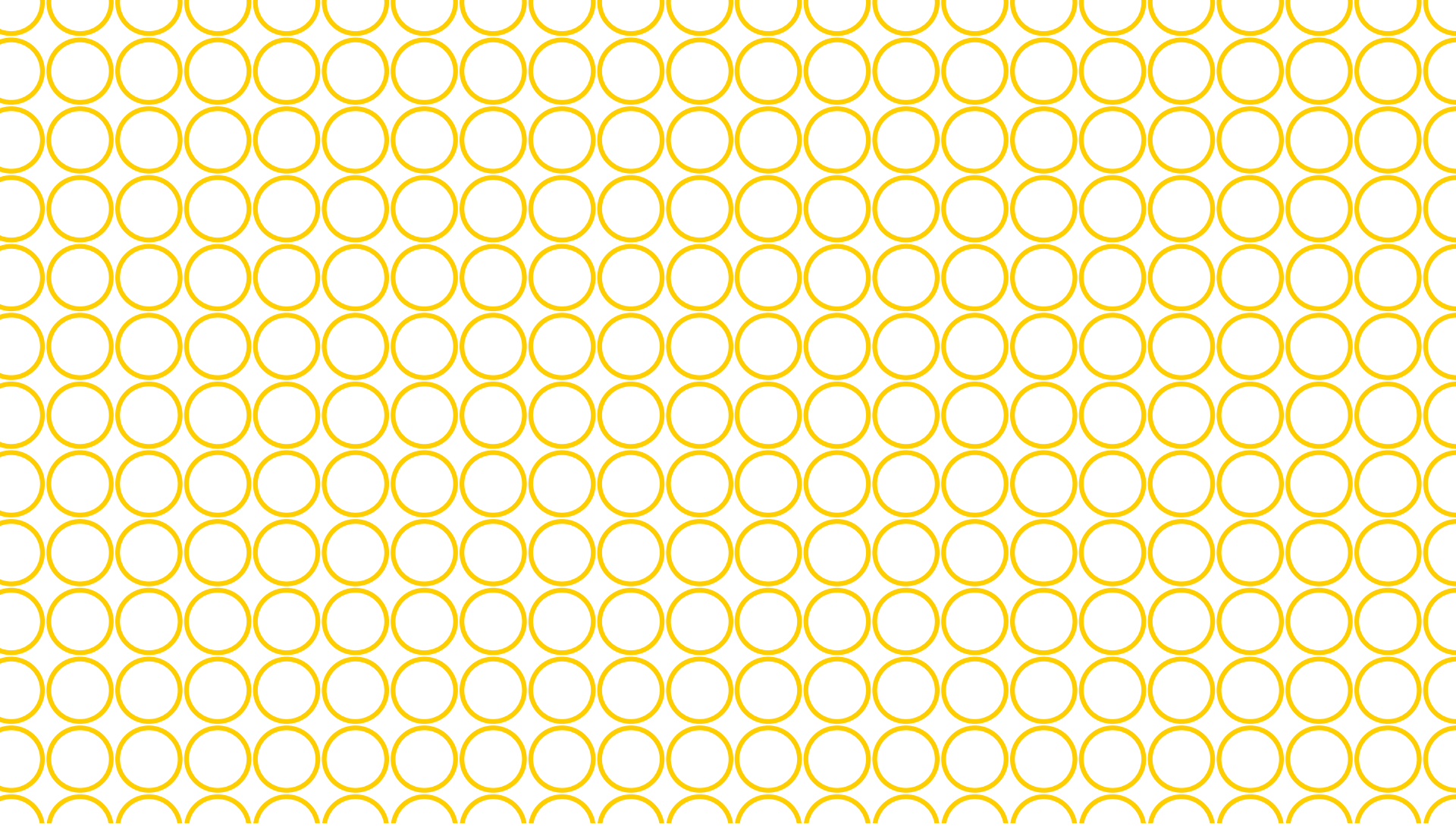
Gebruik dit niet wanneer uw JavaScript acties op de pagina wil uitvoeren vooraleer de pagina volledig geladen is

Content zal reflowen wanneer je in uw uitgesteld JavaScript de DOM gaat aanpassen

Google biedt eigenlijk een andere oplossing aan

http://www.feedthebot.com/pagespeed/defer-loading-javascript.html

```
<script type="text/javascript">
        function downloadJSAtOnload() {
            var element = document.createElement("script");
            element.src = "defer.js";
            document.body.appendChild(element);
        }
        if (window.addEventListener)
            window.addEventListener("load"
                                    , downloadJSAtOnload
                                    , false);
        else if (window.attachEvent)
            window.attachEvent("onload", downloadJSAtOnload);
        else window.onload = downloadJSAtOnload;
    </script>
</body>
```

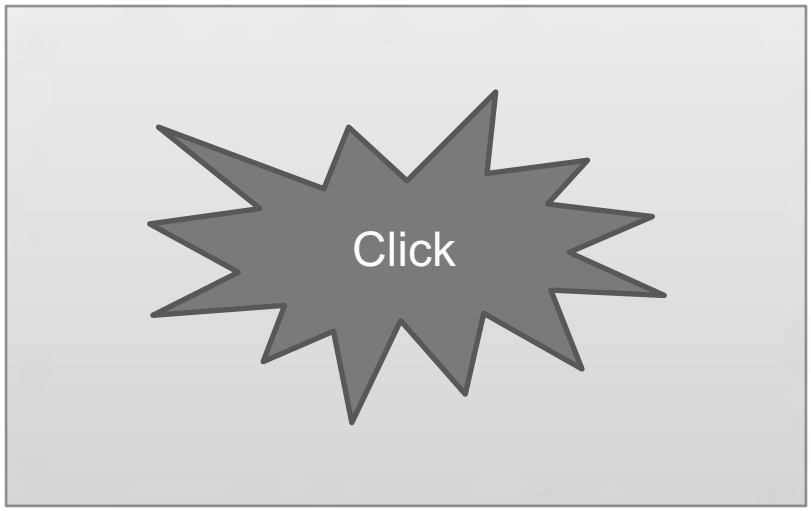


DE UI THREAD

Browser UI Thread is verantwoordelijk voor zowel de UI updates als de uitvoering van de JavaScript.

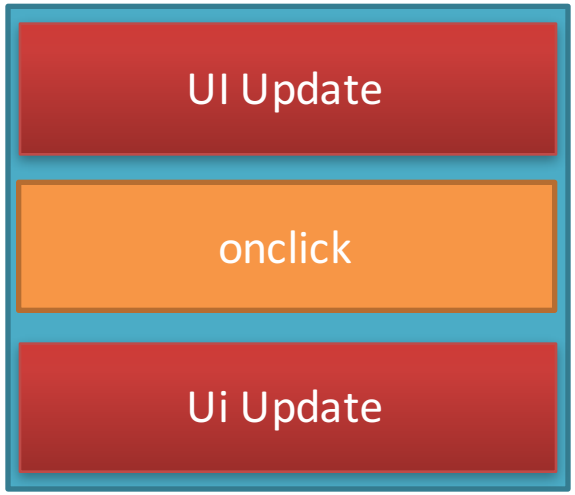
Er wordt maar 1 van de 2 per keer uitgevoerd

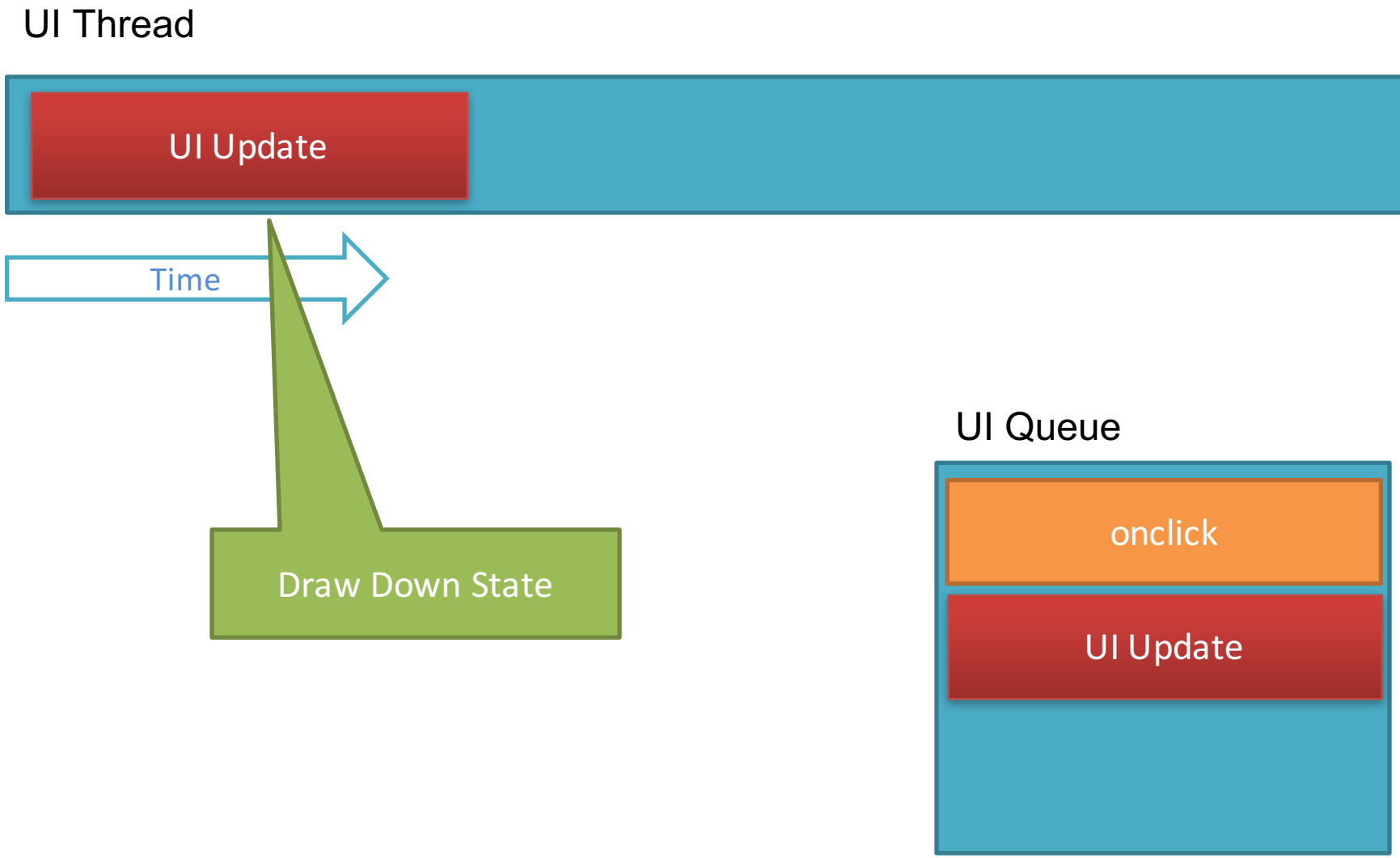
Wanneer de UI bezig is, worden de nieuwe acties in een queue geplaatst

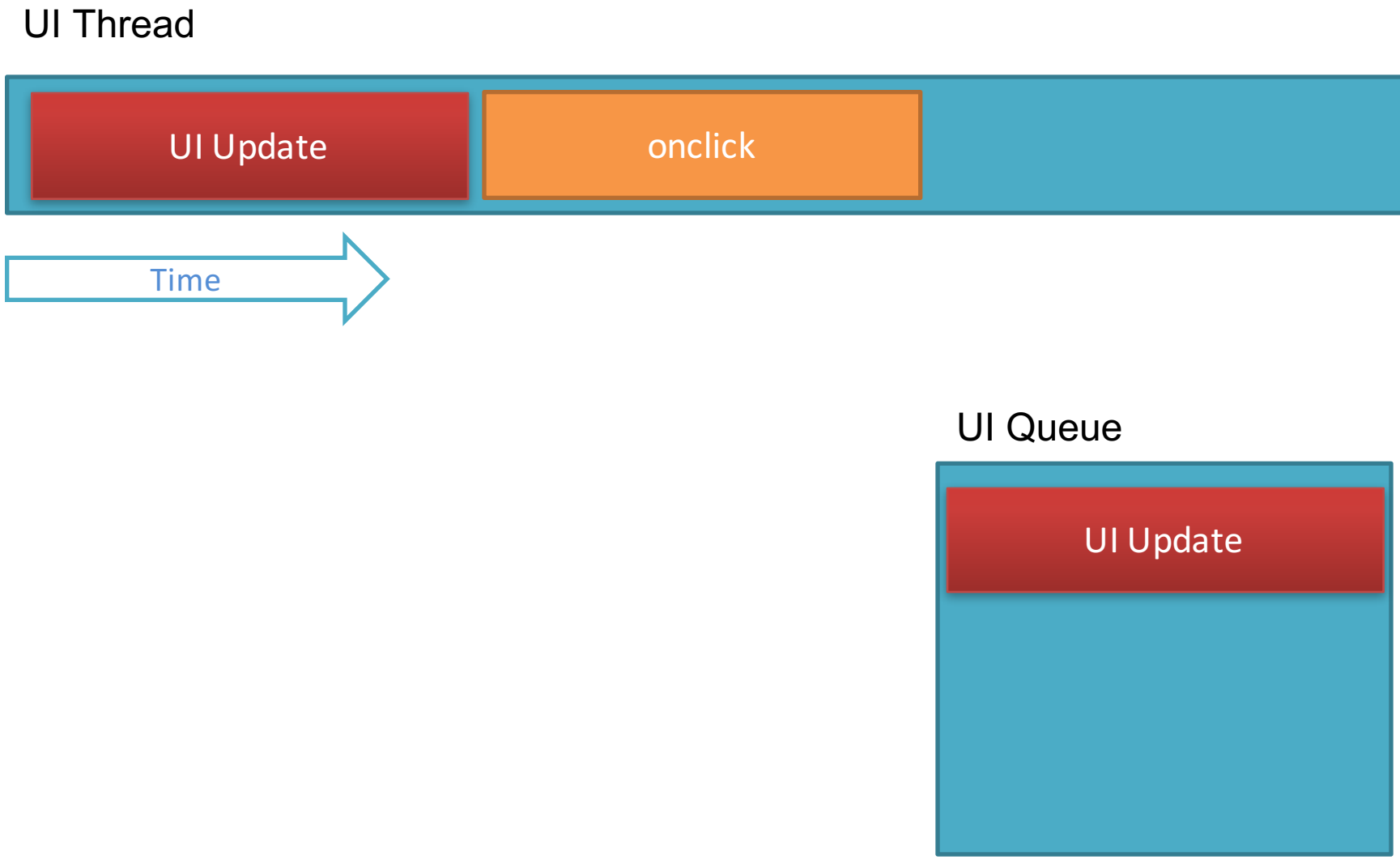

Click

UI Thread

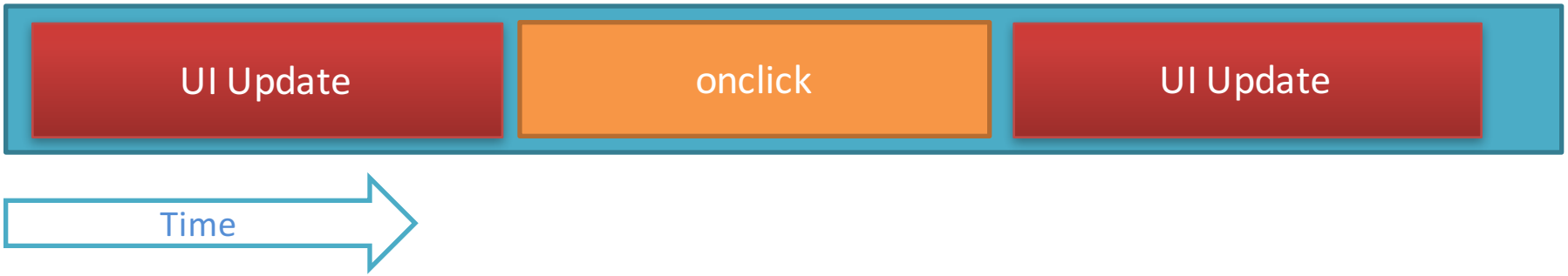
UI Queue




UI Thread
UI Update
Time
Draw Down State
UI Queue
onclick
UI Update

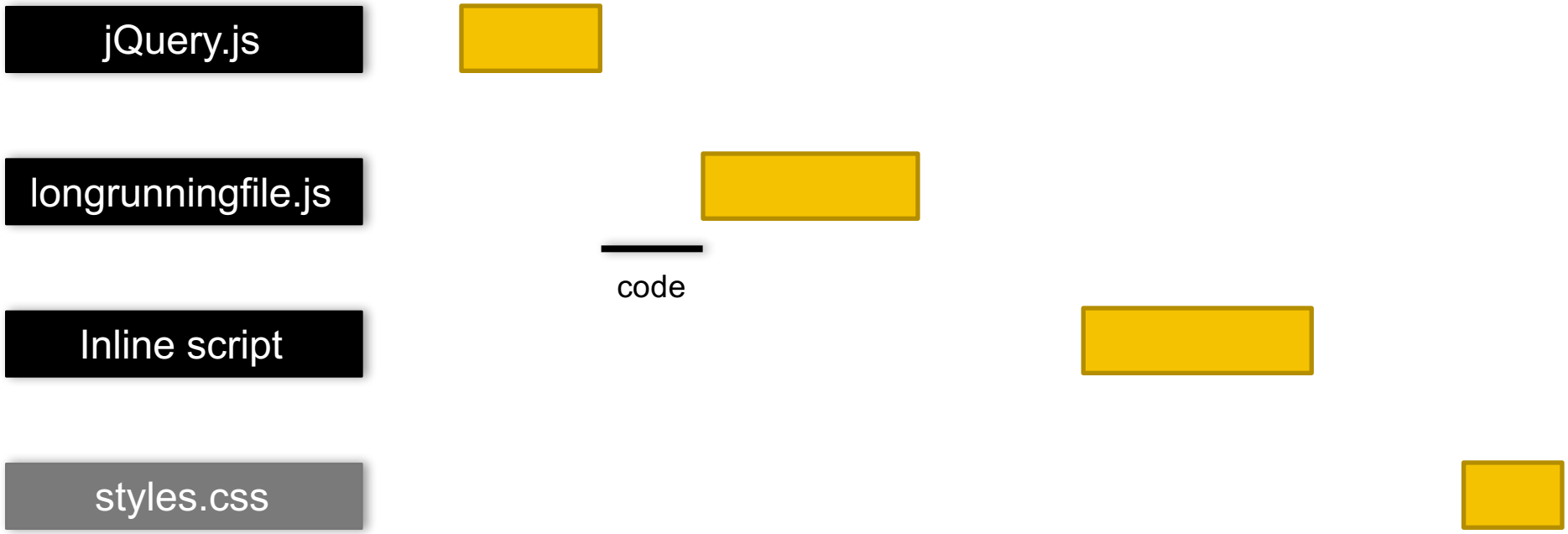

UI Thread
UI Update
onclick
Time
UI Queue
UI Update

UI Thread



UI Queue

Hoe langer JavaScript nodig heeft om uit te voeren, hoe meer unresponsive uw applicatie is


jQuery.js
longrunningfile.js
code
Inline script
styles.css

Web Workers

W3C Candidate Recommendation

Web Workers

W3C Candidate Recommendation 01 May 2012

This Version:
http://www.w3.org/TR/2012/CR-workers-20120501/
Latest Published Version:
http://www.w3.org/TR/workers/
Latest Editor's Draft:
http://dev.w3.org/html5/workers/
Previous Versions:
http://www.w3.org/TR/2012/WD-workers-20120313/
http://www.w3.org/TR/2011/WD-workers-20110901/
http://www.w3.org/TR/2011/WD-workers-20110310/
http://www.w3.org/TR/2011/WD-workers-20110208/
http://www.w3.org/TR/2009/WD-workers-20091029/
http://www.w3.org/TR/2009/WD-workers-20090423/
Editor:
Ian Hickson, Google, Inc.

Web Workers

API voor background scripts in web applications

Voer zaken uit op de background thread zodat uw applicatie niet blokkeert

Waarom

Number crunching

Communicatie met externe sources

Alles wat de UI thread belast

Soorten Web Workers

Dedicated worker

gelinkt aan de browser die de worker gestart heeft

Shared Worker

runt in de background

Nog in geen enkele browser geimplementeerd

Beperkingen

Geen toegang tot de DOM

Geen toegang tot de Window

Geen toegang tot de host page

Dit wil ook zeggen dat je je favoriete framework niet zal kunnen gebruiken.

Meestal zijn die afhankelijk van het window object.

Wat kan je wel

Core JavaScript

JavaScript OO Programming

- navigator: appName, appVersion, userAgent
- Location: href
- Timers: setTimeout, setInterval
- xmlHttpRequest

1. Maak een worker aan

```
worker = new Worker("js/ajaxworker.js")
```

2. Voozie callback methodes

```
worker.onmessage = messageHandler;
worker.onerror = errorHandler;
```

3. Zet de worker aan het werk

```
worker.postMessage("zeg iets");
```

4. Maak een listener aan

```
addEventListener("message", messageHandler, true);
```

5. Laat weten wat het resultaat is

```
function messageHandler(e) {
      if (e.data) {
            postMessage("result" + e.data);
      }
}
```