

IMG IS EVERYTHING

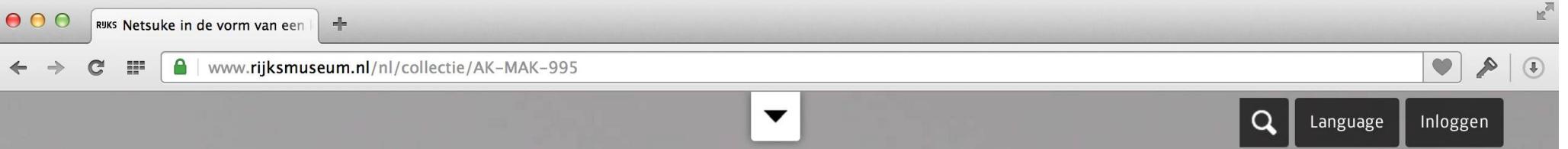
FRONTEND DEVELOPMENT LES 6

VORIGE WEEK

- De principes van het web
- Multiscreen
- Responsive webdesign
- Mediaqueries
- Mobile first

Even heel kort samenvatten waar we het ook al weer over hebben gehad vorige week. Niet te lang bij stilstaan want er is heel veel materiaal te behandelen. Bovendien gaan we dieper in op bepaalde zaken.

PLAATJES



Een van de eerste problemen waar we tegenaan liepen met responsive design was plaatjes. Hoe ga je er mee om dat een plaatje op sommige apparaten gigantisch moet zijn



Netsuke in de vorm van een konijn, Ranichi, 1790 - 1800

♥ x 19





en op andere juist klein...

```
 EN SRCSET

Bovenstaande use cases zijn reëel. Je wil graag dat dit op de client, zonder JavaScript wordt opgelost. Browsers kunnen dit prima zelf. Vandaar dat er twee nieuwe oplossingen zijn.

# SRCSET

```

```

De nieuwe syntax van de sourceset + sizes attributen. We gaan zo naar de details kijken.

# SRCSET

```

```

Laad eerst het kleine plaatje. Hiermee laadt je geen onnodige pixels voor apparaten die het srcset-attribuut niet begrijpen.

# SRCSET

```
<img ...
 srcset="large.jpg 1024w,
 medium.jpg 640w,
 small.jpg 320w"
 ... >
```

In de srcset geef je aan welke plaatjes er zijn en hoe breed ze zijn. Als plaatjes op 100% van de viewport breedte getoond worden is dit genoeg: de browser kiest nu het plaatje wat het beste past, gebaseerd op schermgrootte én resolutie.

# SRCSET

```
<img ...
 sizes="(min-width: 36em) 33.3vw,
 100vw" ... >
```

Als de plaatjes niet altijd op 100vw getoond worden kan je dat aangeven in het sizes attribuut. Bijvoorbeeld "50vw" geeft aan dat plaatjes altijd 50% van de viewport breedte zijn. Je kan hier ook meerdere mediaqueries invoeren. Dan doe je iets als "(min-width: 36em) 33.3vw, 100vw". Hier staat. Als het scherm breder is dan 36em, dan is het plaatje 33.33vw. Anders is het 100vw. Je kunt hier zo veel waardes invoeren als je maar nodig hebt. De laatste is dus de default. Het is dan telkens [media query] [length], [media query] [length] ... etc

# WAT LOST SRCSET OP?

- Breedte van het scherm
- Breedte van het *plaatje*
- Resolutie
- Netwerksnelheid
- Netwerk*prijs*

Deze use cases worden allemaal opgelost met deze twee nieuwe attributen.  
Leg even uit hoezo.

# WAT LOST SRCSET NIET OP?

- Hoogte van het scherm
- Hoogte van het *plaatje*
- Art Direction

Op dit moment kan er nog geen hoogte aangegeven worden in het sizes attribuut. Dat zou je wel willen. Op deze site bijvoorbeeld zijn de plaatjes maximaal 90vh – <http://vvg.gr/im> – hier kan nog niks mee, in de toekomst wellicht wel.

# HOE LOS JE ART DIRECTION OP?

Srcset + sizes is genoeg voor 90% van de gevallen. Voor die gevallen waarin je wel verschillende plaatjes aan verschillende resoluties wilt tonen is er meer nodig. Het Picture element.

# <PICTURE>

```
<picture>
 <source media="(min-width: 45em)" srcset="large.jpg">
 <source media="(min-width: 32em)" srcset="med.jpg">

</picture>
```

Zo los je dat op, op zich. Eens kijken naar de details:

# <PICTURE>

```
<picture>
 ...
</picture>
```

een picture is een lege huls. In het picture element kan je een aantal specifieke dingen stoppen.

# <PICTURE>

```
<source media="(min-width: 45em)" srcset="large.jpg">
<source media="(min-width: 32em)" srcset="med.jpg">
```

In het source element geef je een mediaquery aan, plus een srcset waarin de (mogelijke) bron(nen) staan. Hiermee kan je dus

# <PICTURE>

```
<source media="(min-width: 45em)" srcset="large.jpg, large-hd.jpg"
 <source media="(min-width: 32em)" srcset="med.jpg">
```

Met het 2x ding in de eerste srcset geef je aan dat een plaatje bedoeld is voor retina schermen. Hoei, dat is die use case weer! Je kan ook het sizes attribuut in het source-element gebruiken. Ruimte voor alle mogelijke complexiteit dus. Behalve hoogte, dat kan niet.

# <PICTURE>

```
<
```

Tot slot staat er nog een image element in. Als fallback voor als de picture- en source-elementen niet ondersteund worden.

# FEATURE DETECTION

Met het picture element detecteer je eigenschappen van het scherm én eigenschappen van het plaatje zelf. We kunnen meer dingen detecteren. Zoals bijvoorbeeld CSS properties.

# @SUPPORTS

```
body {
 background: green;
}
@supports (transform-origin: 5% 5%) {
 body {
 background: papayawhip;
 }
}
```

Als transform-origin: 5% 5% niet ondersteund wordt dan wordt de body groen. Anders is ie papayawhip. Wat zou er gebeuren als @supports niet wordt ondersteund? Precies. Dan blijft de background:green. Eens zien of er iemand een nuttige use case kan verzinnen.

# ANDERE FEATURES

```
if('querySelector' in document
 && 'localStorage' in window
 && 'addEventListener' in window) {
 // bootstrap the javascript application
}
```

Er zijn veel features die je zou willen detecteren. Bijvoorbeeld geolocation: als dat er wel is kan je er bijvoorbeeld voor kiezen om het standaard lokatieformulier te vervangen door iets slimmers. Je kan het ook, zoals de BBC gebruiken om te zien of de JavaScript engine van de browser wel slim genoeg is voor de JavaScript die er gebruikt wordt. Zo niet, dan krijg je de simpelere versie, zo wel, dan krijg je de volle laag met features.

# FEATURE VS DEVICE DETECTION

Kennen mensen voor en nadelen van device detection? Bijvoorbeeld: in iOS 7 zat een hele vervelende bug met de vh unit. Door te detecteren of iemand mobile safari gebruikt kan je een alternatieve oplossing aanbieden. Die bug is opgelost in iOS 8. De alternatieve oplossing wordt nog steeds aangeboden als je device detection gebruikt. Als je had getest of het probleem optreedt dan bied je het alternatief alleen aan als het echt nodig is. ga hier, indien er tijd is, vooral even dieper op in. De term "Progressive enhancement" verwacht ik wel te horen in dit verhaal.