Scroll-events – aktivér scripts på bestemte y-koordinater når der scrolles

Når brugeren scroller ned ad siden udløses der hele tiden nye y-koordinater i browseren. Med javascript kan vi „måle“ en række nyttige forhold ved scroll i browseren, fx hvor langt ned ad siden der er scrollet? Hvad er vindueshøjden? dokumenthøjden etc:

1

$(window).scrollTop();
Antal pixels der er scrollet nedad i dokumentet (og som derfor ligger ovenfor browservinduet) , opdateres løbende

$(window).height();
Browservinduets højde, skal opdateres ved ændring af browservinduets størrelse

$(document).height();
Html-dokumentets totale højde

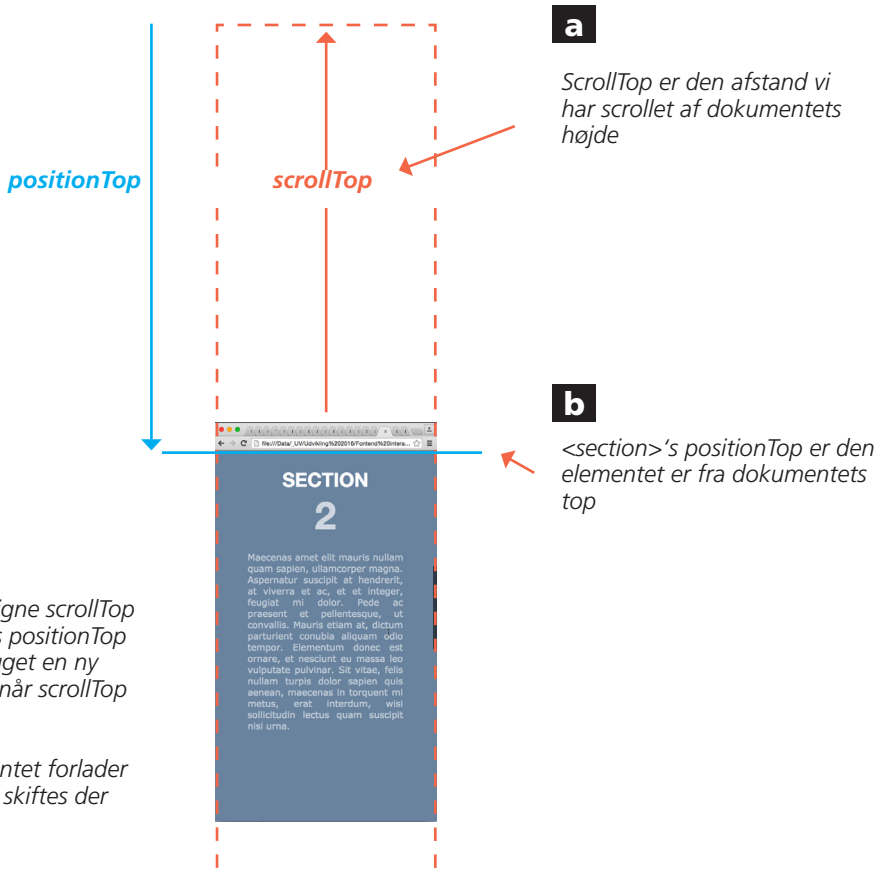
$('section').position().top
Elementets afstand fra dets parents top

$('section'').offset().top
Elementets afstand fra dokumentets top

2 Eksempel med side der ændrer farve når der scrolles

Siden her demonstrer hvordan man løbende kan „læse“ et html-dokuments lodrette scroll og bruge den info til at skifte farve på <body> tagget, eftehånden som nye elementer dukker op på skærmen:

Efterhånden som der scrolles nedad kommer en række <section> ind i browservinduet, ved konstant at måle scrollTop og sammenligne med <section>'s placering i dokumentet kan der sættes regler for om der skal skiftes farve.



javascriptet

Javascriptet har en række foruddefinerede farver, der lægges i variabler. Ellers består det af en „måling" af hhv. scrollTop og positionTop, der udføres når der scrolles (on scroll) og som sammenlignes for hver section i en if-sætning:

```
var farve_1 = "#4c6680";
var farve_2 = "#a0b8cf";
var farve_3 = "#e07f21";
var farve_4 = "#d54215";
$(window).on("scroll touchmove", function() {

if ($(document).scrollTop() >= $("header+section").position().top) {
   $('body').css({'background': farve_1}); };

if ($(document).scrollTop() > $("section:nth-child(3)").position().top) {
   $('body').css({'background', farve_2});  };

if ($(document).scrollTop() > $("section:nth-child(4)").position().top) {
   $('body').css({'background', farve_3});  };

if ($(document).scrollTop() > $("section:last-child").position().top) {
   $('body').css({'background', farve_4});  };
            });
```

HTML-koden

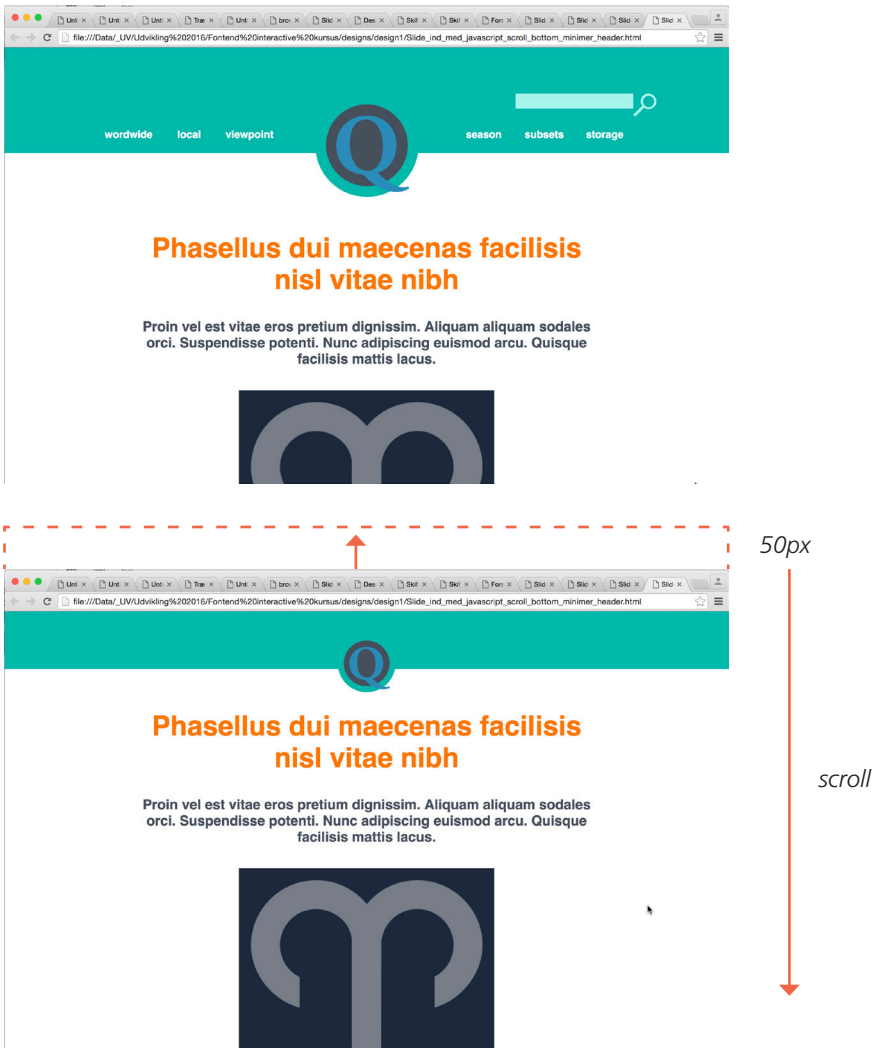
Html-koden har denne opbygning:

```
<body>
        <header><img src="billeder/logo.png" alt="logo"></header>
        <section>
            <article><h2>Section<br><span>1</span></h2>
                <p>Lorem ipsum ...
                </p>
            </article>
        </section>

        <section>
            <article><h2>Section<br><span>2</span></h2>
                <p>Maecenas amet ....
                </p>
            </article>
        </section>

        <section>
            <article><h2>Section<br><span>3</span></h2>
                <p>Porta purus ac, ....
                </p>
            </article>
        </section>
        <section>
            <article><h2>Section<br><span>4</span></h2>
                <p>Metus non ipsum ....
                </p>
            </article>
        </section>

    </body>
```

Eksempel med skalérbar head-del

På denne side minimeres en fixed positioned header når der scrolles ned ad siden. Det sker ved at dens højde skaleres og udvalgte elementer fader ud. Og scroller man opad igen reetableres det oprindelige design:

1



Vi arbejder med en fastlagt grænseværdi for hvornår head-delen skal minimeres – i eksemplet er der tale om 50 pixels, så når siden er scrollet 50 pixels overføres nye css-regler med addClass

2 **Javascriptet, først en række variabler – herefter scroll-funktionen:**

```
var scrollTop = $(window).scrollTop();
    //antal pixels der er scrollet nedad i dokumnetet (og som derfor ligger ovenfor browservinduet, skal opdateres løbende
var height = $(window).height(); // browservinduets højde, skal opdateres ved ændring af browservinduets størrelse
var pageHeight = $(document).height(); // htmldokumentets totale højde
var bundLimit = ($('footer').offset().top - height);// det pixeltal  der skal trigge funktionen
var undgaa_trigger = false;

 $(document).on('scroll',function(e) {
   scrollTop = $(window).scrollTop();  //console.log("vi er scrollet: " + scrollTop);
            if(scrollTop >= 50) {
                    $('header').addClass('minimer');  //    console.log('minimer');
                }
            }
```

3 Css'en:

Javascriptet tildeler og fjerner class'en „minimer" til <header>. Men der er også oprettet en række separate css-regler til „børn" af header, som er betinget af at header har fået class'en „minimer", fx „header.minimer nav" og „header.minimer div". På den måde sætter det mange flere transitions i gang end blot headers animation, når header får tildelt sin klasse.

```
header {
        background : rgba(0, 185, 172, 1);
        width : 100%;
        height : 183px;
        text-align: center;
        position:fixed;
        z-index: 10;
        top:0;
        transition: .5s  ease-in-out;

.minimer {
        -webkit-transform: translateY(-40px);
        height:140px; }

header div {
        position: absolute;
        top:80px;
        right:0;
        transition:.5s;
        -webkit-opacity:1}

header.minimer div {
        -webkit-opacity:0;
        top:60px}
```

```
header nav {
        -webkit-opacity:1;
        -webkit-transition: .5s  ; }

    header.minimer nav {
        -webkit-opacity:0; }


header nav li {
        text-align: center;
        width:10%;
        display:inline-block;
        list-style: none;
        -webkit-transform:translateY(140px);
        -webkit-transition:.5s ;
          }

header.minimer nav li {
        -webkit-transform:translateY(100px); }
```

Html-strukturen i header

For den gode ordens skyld, og for at forklare css'en – er her html'en i header

```
<body>
        <header>
            <article>
                <img src="images/logo.svg">
                <div><input type="text" > <img  src="images/lup.svg"> </div>
                <nav>
                    <ul>
                        <li><a href="#">wordwide</a></li>
                        <li><a href="#">local</a></li>
                        <li><a href="#">viewpoint</a></li>
                        <li><a href="#">season</a></li>
                        <li><a href="#">subsets</a></li>
                        <li><a href="#">storage</a></li>
                    </ul>
                </nav>
            </article>
        </header>
```

4 Slide-ind boxen, javascript og css

Slide-ind boxen skal have sinde egne variabler der styrer ved hvilken afstand fra bunden den skal bevæge sig ind i browseren. Dens animation er styret af en css-transition, der ligger i class'en „slide_pop_ind":

```
var bundLimit = ($('footer').offset().top - height);     // det pixeltal der sætter grænsen for hvornår slide-pop-ind funktionen skal trigges
```

Selv scroll-funktionen udvides med disse linjer:

```
if(scrollTop  >= bundLimit) {
    //console.log(scrollTop + "= scrollTop " + bundLimit);
    $("#pop-ind").addClass('slide_pop_ind');
                    }

if(scrollTop  <= bundLimit) {
    $("#pop-ind").removeClass('slide_pop_ind');
                    }
```

Css'en der styrer boxens adfærd ser sådan ud:

```
.slide_pop_ind {
    -webkit-transform:translateX(-50vw) !important;
    -webkit-opacity:1 !important;
}
```

*De to **!important** tilføjelser er skrevet af specificity hensyn, så klassens regler får "vægt" i forhold til reglerne med selektoren **div#pop-ind...***

Alternativt skulle selektoren for klassen se sådan ud:
div#pop-ind.slide_pop_ind

```
div#pop-ind {
    background : rgba(72, 81, 94, 1);
    border-style : Solid;
    border-color : rgba(124, 174, 194, 1);
    border-width : 4px;
    width : 374px;
    height : 245px;
    position:fixed;
    top:400px;
    right:-450px;
    padding:30px;
    -webkit-opacity:0;
    -webkit-transform: translateX(0);
    -webkit-box-shadow: 10px 10px 30px rgba(0,0,0,.2);
    -webkit-transition: .5s ease-in-out; }
```

5 Slip af med slide-ind boxen...

Slide-ind boxen kan være virkelig irriterende – så brugeren skal have mulighed for at sige nej taj, vælge den fra og ikke se den igen så længe han befinder sig på siden.

Derfor udstyres den med en „knap“ eller et <a>-tag, der ved et klik sørger for at:

1) fratage slide-ind-boksen klassen „.slide-pop-in“, så den animerer ud af syne...

2) opretter en variabel, der kan „huske“ brugerens fravalg af slide-ind boksen

Luk-knappen er et a-tag med et et baggrundsbillede, absolut positioneret lidt ud af boksen...

Herefter skal scriptet, der styrer boksens adfærd, ikke blot checke afstanden til bunden af siden – og om slide-ind boksen skal animeres ind – men også checke variablen for om brugeren har fravalgt at se boksen igen!

Luk-knappens script

Til at rumme brugerens valg oprettes i scriptets top en variabel, „undgaa-trigger“, der som start tildeles værdien „false“:

```
var undgaa_trigger = false;
```

Når der klikkes på luk-knappen fratager scriptet på den ene side slide-ind-boksen klassen „.slide-pop-in“ med removeClass. På den anden side tildeles variablen „**undgaa_trigger**“ værdien „**true**“:

```
$("#pop-ind a").on('click', function(e) {
   e.preventDefault();
   $("#pop-ind").removeClass('slide_pop_ind');
   undgaa_trigger = true;
                });
```

if-sætning der checker om to ting er sande

Der er nu brug for en if-sætning der både kan checke brugerens valg i variablen „undgaa_trigger“ og bedømme afstanden til slutningen af siden. Hvis en if-sætning skal checke flere ting på én gang har du brug for at sige noget i retning af:

```
if( det ene == sandt og det andet andet == sandt ) { så udfør disse handlinger }
```

Til det formål findes en såkaldt „logisk operator“, der scriptes med tegnene „**&&**“. I forbindelse med scroll-funktionen skal koden således checke afstanden til bunden OG værdien af variablen „undgaa-trigger“:

If-sætningen der allerede trigger #pop-ind-boksens animation ind på siden, rettes derfor til følgende:

```
if(scrollTop  >= bundLimit && undgaa_trigger == false) {
     $("#pop-ind").addClass('slide_pop_ind');
                  }
```

Det, der står her er: hvis scroll overskrider grænsen mod sidens bund –og brugeren ikke har valgt boksen fra, så slider vi #pop-ind ind på siden...

Men hvis brugeren omvendt havde lukket boksen og variablen undgaa_trigger havde værdien „true“, ville if-sætingens handlingsblok aldrig udføres, og boksen villle ikke blive animeret ind.

6 Det samlede javascript

Her er det samlede javascript, for overblikkets skyld:

```
<script>

            $(function() {
                var scrollTop = $(window).scrollTop();
                var height = $(window).height();
                var pageHeight = $(document).height();
                var bundLimit = ($('footer').offset().top - height);
                var undgaa_trigger = false;


                $(document).on('scroll',function(e) {
                    scrollTop = $(window).scrollTop();

                     if(scrollTop >= 50) {
                        $('header').addClass('minimer');
                             }

                     if(scrollTop <= 50) {
                        $('header').removeClass('minimer');
                             }


                     if(scrollTop  >= bundLimit && undgaa_trigger == false) {
                        $("#pop-ind").addClass('slide_pop_ind');
                             }

                     if(scrollTop  <= bundLimit) {
                        $("#pop-ind").removeClass('slide_pop_ind');
                             }
                });


                $("#pop-ind a").on('click', function(e) {
                    e.preventDefault();
                    $("#pop-ind").removeClass('slide_pop_ind');
                    undgaa_trigger = true;
                });

            })

</script>
```

Eksempel med scroll der trigger animation af udvalgte elementer ned ad siden

1

På denne side kommer udvalgte elementer til syne ved at animeres ind på siden efterhånden som de kommer indenfor browservinduets flade.

Elementerne der skal animeres er som princip „børn“ af et containerelement, der tilknyttes en klasse. Det kan fx være billeder, <img>'s der er børn af en <figure> som har klassen – eller det kan være <li>, hvor det er -tagget der har klassen.

Er der flere søskende-elementer, som fx <li> i en <ul>, vil de animeres en ad gangen, idet de hver især får et „delay“ hvis forsinkelse vokser proportionalt med elementets position i søskendeflokken...

Det er svært at illustrere – men elementerne ankommer animeret ind på siden... en ad gangen efter tur

2

Lidt forklaringer til scriptet

De elementer der skal animeres placeres i en „container“, som får tildelt klassen „.animation-element“. Scriptet vil herefter vælge samtlige child-elementer indeni klassen med en jquery-selektor der ser sådan ud:

```
$('.animation-element > *')
```

I scriptet lægges disse i en variabel, så browseren kun skal ulejliges én gang:

```
var $animation_elements = $('.animation-element > *');
```

Som udgangspunkt gøres alle childelementer usynlige og skaleres lidt op med csstil delt af javascript:

```
$animation_elements.css({ '-webkit-opacity' : '0', '-webkit-transform' : 'scale(1.1)' });
```

Det er childelementernes position, der afgør om de skal animeres ind på siden. Derfor udstyres de alle med en funktion, der sætter dem i stand til at undersøge dette.

Når siden scrolles eller skaleres køres funktionen:

```
$window.on('scroll resize',  function() {
```

Først oprettes en række variabler, der letter scriptets kode efterfølgende:

```
var window_height = $window.height();
var window_top_position = $window.scrollTop();
var window_bottom_position = (window_top_position + window_height);
```

Med „each" udstyres hvert fundet element med en funktion der er bliver den „adfærd" – dvs animation og evne til at afgøre hvornår denne skal aktiveres:

```
$.each($animation_elements, function() {
```

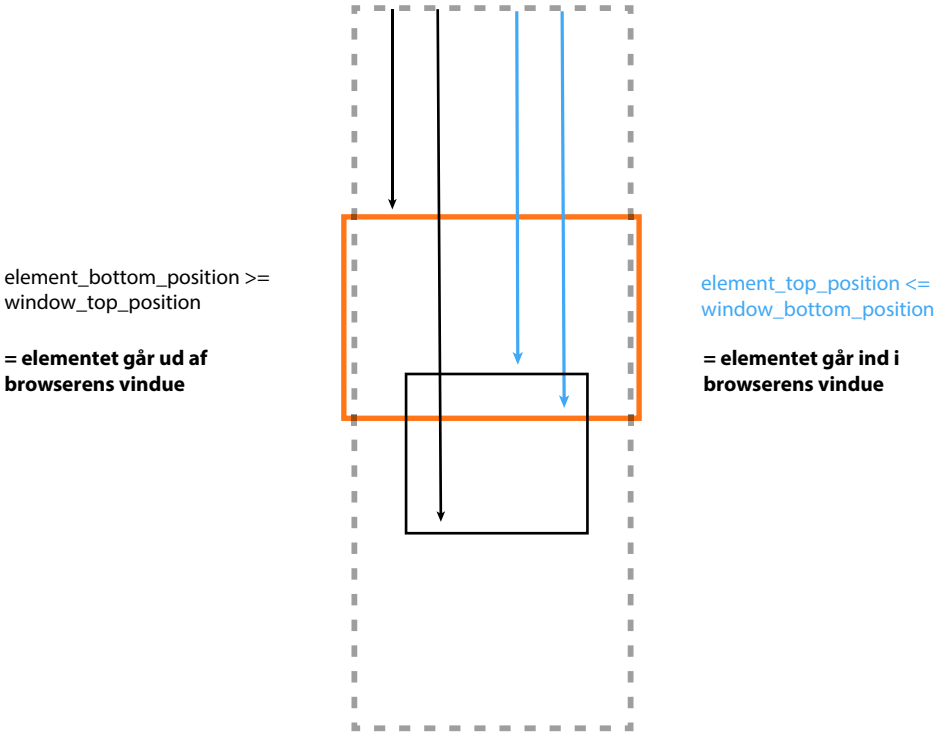
Hvert element „husker" sin højde, sin afstand til toppen af websiden og „højden" – dvs. y-koordinatet, for sin underkant:

```
var element_height = $(this).outerHeight();
var element_top_position = $(this).offset().top;
var element_bottom_position = (element_top_position + element_height);
```

Denne if-sætning checker om elementet er indenfor browservinduet:

```
if ((element_bottom_position >= window_top_position) &&
(element_top_position <= window_bottom_position)) {
```

Det der sker kan illustreres sådan her:



Herefter tildeles css'en med en transition til opacity(1) og scale (1), så elementet animeres ind og skaleres ned til 100%.

Når transition er færdig-afviklet fjernes klassen „animation element", så animationen ikke udføres igen, selvom brugeren scroller op og ned ad siden

```
$(this).css({'-webkit-transition':'800ms' + $(this).index()*500 +'ms ease-in-out',
            '-webkit-opacity' : '1','-webkit-transform': 'scale(1)' })
                    .on('webkitTransitionEnd', function(){
                    $(this).parent().removeClass('animation-element');})
            }
```

Hvis du ønsker at elementerne skal animere hver gang de passerer browservinduet, skal ".on("webkitTransitionEnd"...") fjernes og den udkommenterede "else" linje aktiveres, så elementerne gøres usynlige igen når de forlader browservinduet:

```
/* else {
  $(this).css({'-webkit-opacity' : '0','-webkit-transform' : 'scale(1.05)'});
          } */
```

3 Scriptet

```
$(function(){
 var $animation_elements = $('.animation-element > *');

 var $window = $(window);
 $animation_elements.css({ '-webkit-opacity' : '0', '-webkit-transform' :
'scale(1.1)' });

 $window.on('scroll resize',  function() {
 var window_height = $window.height();
 var window_top_position = $window.scrollTop();
 var window_bottom_position = (window_top_position + window_height);

 $.each($animation_elements, function() {

 var element_height = $(this).outerHeight();
 var element_top_position = $(this).offset().top;
 var element_bottom_position = (element_top_position + element_height);

 if ((element_bottom_position >= window_top_position) &&
 (element_top_position <= window_bottom_position)) {
 $(this).css({
 '-webkit-transition' : '800ms ' + $(this).index()*500 +'ms ease-in-out',
 '-webkit-opacity' : '1','-webkit-transform' : 'scale(1)' })
 .on("webkitTransitionEnd", function(){
 $(this).parent().removeClass('animation-element');})
                    }

// hvis du ønsker at elementerne skal animere hver gang de passerer browservinduet, skal ".on("webkitTransi-
tionEnd"..." fjernes og denne "else" linje aktiveres:

                    /*else {
                      $(this).css({'-webkit-opacity' : '0','-webkit-transform' :
'scale(1.05)'});
                    } */
                });
            });
        });
```