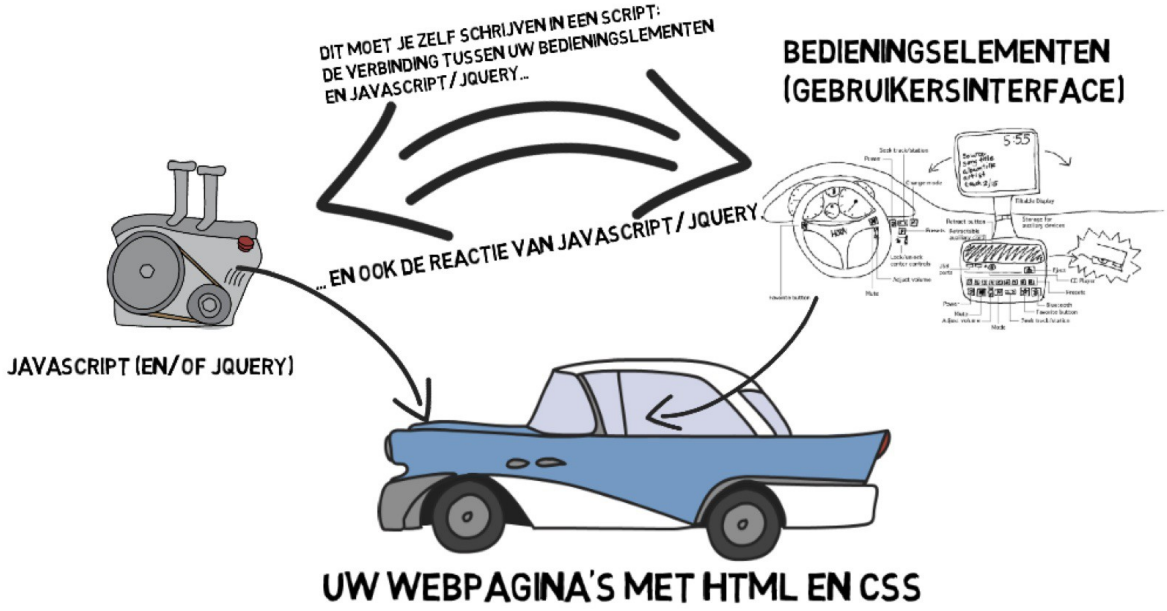
Javascript

Je hebt een site gebouwd met HTML en CSS gebruikt voor de opmaak. Toch ontbreekt er nog wel wat. Je site is mooi en goed opgebouwd, maar mist interactiviteit met de gebruiker. De gebruiker kan op dit moment enkel naar je pagina's kijken en de informatie lezen, maar er zit voor de rest weinig interactie in. Het lijkt op een mooie auto zonder motor. Daar kan je verandering in brengen.

> *Vergelijk het even opnieuw met een auto. In een auto zit een motor en die motor bestaat op zijn beurt uit een hoop kleinere modules. Er zit een carburator, een pomp, cilinders, een radiator, waterpomp enz. in. Uiteraard is zo'n motor alleen niet voldoende. De gebruiker kan de motor en alle onderdelen hiervan interactief besturen. Wanneer hij aan de sleutel draait, start de motor. Als de bestuurder het ontkoppelingspedaal indrukt, de versnellingsbak in de juiste versnelling plaatst en vervolgens het pedaal zachtjes loslaat, komt de auto in beweging. De pedalen, sleutel, versnellingen... zijn interface-elementen die er voor zorgen dat de gebruiker interactief de motor kan aansturen. Het resultaat is dat de auto in beweging komt, de ruitenwissers beginnen bewegen, de lichten kunnen worden aangezet enz. Tussen die interface-elementen en de onderdelen van de motor liggen verbindingen, kabels, slangen enz. die ervoor zorgen dat alles netjes wordt aangestuurd.*



Javascript is een taal waarmee u **interactiviteit** kan toevoegen aan HTML. Hieronder vind je een kort overzicht van de mogelijkheden van javascript:

- *Aanpassen, toevoegen, klonen, verwijderen... van om het even welk HTML-element in de huidige pagina.*
- *Het aanpassen van inhouden van om het even welk HTML-element.*
- *Het aanpassen van CSS-eigenschappen van de HTML-elementen op de pagina.*
- *Controleren van formulieren.*
- *Het inlezen van externe bestanden vanop dezelfde server.*
- *Manipuleren van interne en externe XML- en JSON-gegevens (in mensentaal: gestructureerde gegevens zoals in bijvoorbeeld Excel staan, openen en tonen)*
- *Tekenen in het HTML5-canvas-element (in mensentaal: tekeningen maken in de webpagina of effecten geven aan foto's)*
- *Opvragen van locatiegegevens (na toelating van de gebruiker).*
- *Bewaren van gegevens in de browser of in een offline databank (o.a. de bekende “cookies”)*
- *Drag and drop binnen en naar buiten de browser (vb. producten slepen naar een “winkelmandje”)*
- *...*

De mogelijkheden van javascript zijn nagenoeg '**onbegrensd**'. Toch kent de taal wel wat **beperkingen**. Voor de *veiligheid* kan je niet zonder meer ingrijpen in het systeem van de gebruiker. Je kan bijvoorbeeld geen bestanden van de computer van de bezoeker verwijderen. Toch wordt in de toekomst wellicht nog meer mogelijk op dit vlak: toegang tot de webcam en de microfoon...

Javascript is een relatief eenvoudig te leren scriptingtaal. **De broncode staat leesbaar in de HTML-code. De code wordt uitgevoerd door de javascript-engine van uw browser**. Bij een echte programmeertaal wordt de code vooraf 'gecompileerd' naar bits en bytes (nullen en enen, weet je wel?) voor de respectieve computerprocessor. Bij javascript is dit niet zo. De code wordt pas gecompileerd en uitgevoerd wanneer de browser de HTML-pagina en gekoppelde scripts inleest vanaf het domein waarop de website staat.

HOE?
Javascript neem je met het <script>-HTML-element op in uw HTML-code. U kan javascriptcode ook in een extern bestand opslaan en dit aan uw pagina linken. Dit js-bestand hoeft zelfs niet op dezelfde server te staan, maar kan ook vanaf een ander domein ingelezen worden.

jQuery maakt het “makkelijk”

jQuery is geen nieuwe taal die u moet onder de knie krijgen. **Het is een manier om sneller javascriptcode te kunnen schrijven die bovendien in alle browsers (ook mobiele) werkt.**

Eerste probleem met javascript:

> Wanneer u code rechtstreeks in javascript schrijft, moet u die taal niet alleen goed onder de knie hebben, maar ook rekening houden met mogelijke verschillen tussen browsers. Zeker oudere versies van Internet Explorer durven wel eens moeilijk doen.

Tweede probleem: veel webontwikkelaars vinden telkens opnieuw het warme water uit:

> Vaak gebruik je als webontwikkelaar dezelfde stukjes javascript keer op keer opnieuw. Veel websites vragen immers dezelfde of gelijkaardige interactiviteit.

Oplossingen voor het eerste en het tweede probleem:

> Een aantal ontwikkelaars hebben daarom stukken herbruikbare code/scripts samengevoegd in een soort van **bibliotheek of framework**. Zo'n bibliotheek bestaat uit een aantal **functies** (=motoren) die u vanuit uw eigen javascriptcode kan oproepen. Dat bespaart u een pak werk.

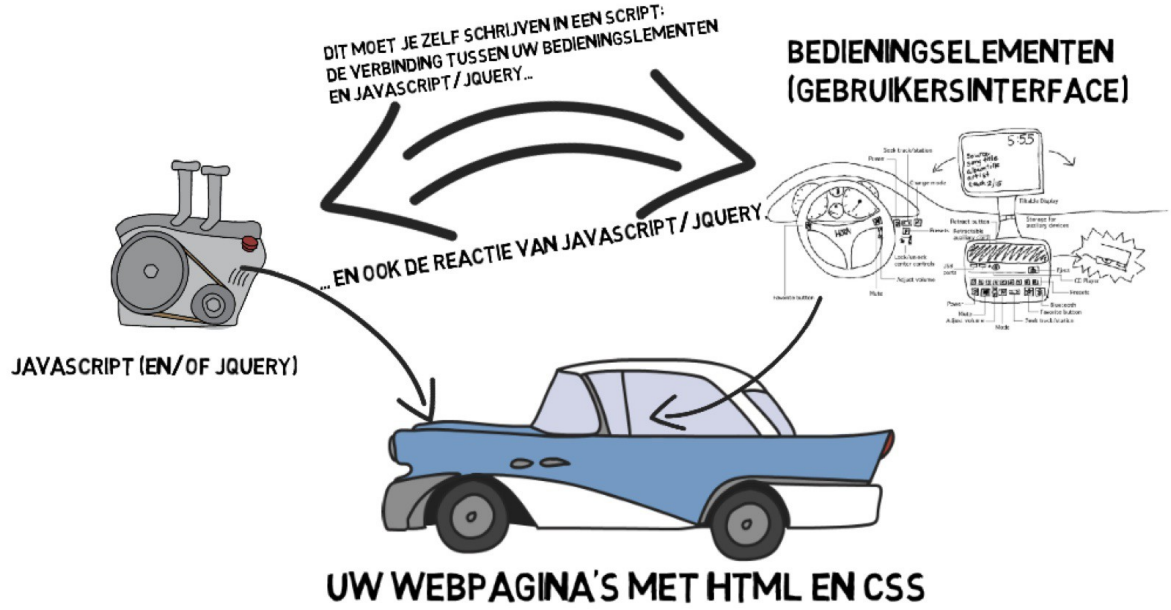
De ultieme eenvoudige oplossing:

> Een aantal frameworks gaat nog verder.
>
> 1. Ze voorzien niet alleen een hoop standaardfuncties,
> 2. maar maken het ook mogelijk om die functies op een heel handige manier aan uw eigen HTML-elementen te koppelen.
> 3. Ze gebruiken een schrijfwijze die heel erg lijkt op die van CSS om elementen te selecteren en er vervolgens een functie op uit te voeren. *Met één lijn code schrijf je bijvoorbeeld een javasriptcode om bij de klik op een knop een bepaald pagina-onderdeel te vervangen door de inhoud van een externe webpagina.*
> 4. Naast een handige manier om elementen te selecteren en een hoop ingebouwde functies, bevatten deze frameworks nog een paar andere interessante mogelijkheden.
>   - *Plugins of insteekmodules*: andere ontwikkelaars kunnen het framework uitbreiden met hun eigen functies en die in de vorm van 'plugins' weer delen met anderen.
>   - *Modules voor de gebruikersinterface:* het framework laat toe om heel snel knoppenbalken, schuifregelaars, 'vensters', accordion-elementen, tabbladen enz. aan een pagina toe te voegen.
>   - *Animaties:* het bouwen van animaties met foto's, lagen en tekst wordt met simpele functies vereenvoudigd.

Het bekendste framework is zonder twijfel **jQuery** van Firefox-ontwikkelaar John Resig.

Howto?

We bekijken nogmaals de tekening:



We moeten...

1. een javascriptmotor aan onze pagina's toevoegen.
2. bedieningselementen toevoegen aan onze webpagina's.
3. Opgelet: sommige dingen gaan ook automatisch zonder dat de gebruiker iets moet doen (vb. bij een ruitenwisser kan een regensensor de ruitenwissers automatisch in beweging zetten)
4. Bepalen welke “actie” gebruiker moet doen vooraleer er iets “gebeurt”.
5. Een verbinding leggen tussen de bedieningselementen en/of de “sensoren” en de motor of een bepaald onderdeel van de motor én bepalen welke “actie” moet uitgevoerd worden.
6. Het eindresultaat tonen aan de gebruiker.

STAP 1: een javascriptmotor aan je webpagina's toevoegen.

Deze stap hoef je eigenlijk in je “project” niet meer te zetten. In de voorbeeldpagina's zit de jQuery-motor reeds in al onze pagina's. Maar het is toch wel belangrijk om te weten hoe je dit doet. Tenslotte ben jij de “garagist”!

Tussen <head> en </head> vind je twee gekoppelde javascript-motoren. Je merkt dat je die niet op je eigen “site” moet zetten. Je kan ze koppelen vanaf de servers van Google:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.11.1/jquery-ui.min.js"></script>
```

Ook zie je dat er ook een stijlbestand is gekoppeld dat voor de correcte werking van bepaalde onderdelen van de jQueryui-motor zorgen. Met de extra motor jQuery UI kan je speciale “interface-elementen” (bedieningselementen zoals knoppen, schuifregelaars, tabbladen...) aan je webpagina's toevoegen.

```
<link rel="stylesheet"
href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.11.1/themes/smoothness/jquery-ui.css" />
```

STAP 2: bedieningselementen op je website

Dankzij jQuery kan je elk onderdeel van je webpagina heel makkelijk interactief maken. Maar het is vaak handig voor de gebruiker om ook duidelijk te laten zien dat hij op een “knop” kan klikken.

Een knop kan je op de volgende manier toevoegen:

```
<button>Lees het volledige artikel.</button>
```

Net zoals in CSS moet je bij jQuery eerst de elementen selecteren waarop je een bepaalde functie of methode wil uitvoeren. Ook de manier waarop je die doet, lijkt erg op die van CSS. Toch beschikt jQuery over nog veel uitgebreidere selectiemogelijkheden.

Je selecteer een element met de functie $();

Hieronder vind je een paar basistechnieken voor het selecteren van elementen:

Alle links op een pagina	$("a")
Alle links die zich in een nav-element bevinden	$("nav a")
Selecteren van alle elementen met een bepaalde klasse. *Bijvoorbeeld: Alle HTML-elementen met de klasse class=”kris”*	$(".kris")
Selecteren van een element op basis van zijn id. Bijvoorbeeld: Het element met als id="contact"	$("#contact")
...	...

Stel dat je op je pagina 2 knoppen toevoegt:

```
<button>Lees het volledige artikel.</button>
<button>Lees het tweede artikel.</button>
```

Dan kan een selectie op deze manier $(“button”) voor problemen zorgen. Je selecteert dan immers alle knoppen. In een auto zou dit er bijvoorbeeld toe leiden dat je ongeacht de knop waarop je drukt steeds de ruitenwissers zo activeren.
Beter is om elke knop een unieke identiteit (id) te geven en de knop op die manier te selecteren:

```
<button id=”BtnLeesArtikel1”>Lees het volledige artikel.</button>
<button id=”BtnLeesArtikel2”>Lees het tweede artikel.</button>
```

Mensen die al jaren met javascript en jQuery werken, zouden het nog anders doen, maar … we proberen het hier wat “eenvoudig” te houden:

Je kan de knoppen nu als volgt selecteren:

```
$(“#BtnLeesArtikel1”)
$(“#BtnLeesArtikel2”)
```

STAP 3: automatische acties zonder interactie (denk aan 'sensoren')

Je kan dingen ook automatisch laten gebeuren met jQuery zonder dat de gebruiker iets doet. Automatisch betekent dat iets meteen begint als de gebruiker de pagina opent in zijn browser of... na een bepaalde tijd.

```
<div id=”banner”>
        DIT IS DE BANNER MET EEN ACHTERGRONDAFBEELDING IN CSS
</div>

<script>
        $(“#banner”).delay(4000).fadeOut();
</script>
```

Met de bovenstaande code zal het element met id=”banner” automatisch vervagen na 4 seconden (4000 staat voor 4000 milliseconden)

Opgelet (zeker lezen): *in principe kan je een script pas starten als alle onderdelen van de pagina (teksten, foto's enz.) zijn geladen. Je zou immers problemen kunnen krijgen als je een afbeelding automatisch wil laten vervagen als die afbeelding nog niet “geladen” is door de browser. Het kan soms immers een paar seconden duren vooraleer een webpagina volledig in de browser van de gebruiker is geladen (vb. door een trage internetverbinding).*

Daarom moet je ervoor zorgen dat je je script pas laat werken als alle onderdelen van de pagina (het “document”) geladen (“ready”) zijn.

```
<script>
        $(document).ready();
</script>
```

Pas dan mag je je eigen code laten uitvoeren:

```
<script>
        $(document).ready(
                function(){
                        //HIER KOMT JE EIGEN CODE
                }
        );
</script>
```

STAP 4 : bepalen welke “acties” een “motor” (function) starten en laten werken

Als je de gewenste elementen hebt geselecteerd, moet je nog bepalen welke gebruikersacties de gewenste motor zullen starten.

Eén van die acties zag je hierboven al nl. $(document).ready(). De actie ready() doet zich voor als alle onderdelen van de pagina geladen zijn.

Opgelet: *je kan code in één script-blok combineren. Alle elementen die je selecteert, mogen allemaal binnen één en hetzelfde scriptblok komen, ongeacht waar ze op de pagina staan.*

```
<script>
	$(document).ready(
		function(){
			$(“#BtnLeesArtikel1”);
			$(“#BtnLeesArtikel2”);
		}
	);
</script>
```

Welke acties kunnen er allemaal “gebeuren”?

Als het regent, doe je je paraplu open. Als je aan de sleutel van je auto draait, start de motor... Zo gaat het ook in javascript. In javascript noemt men dit “events” (=gebeurtenissen). Als er zich een bepaalde gebeurtenis (=event) voordoet, kan je een bepaalde actie uitvoeren.

Tot de belangrijkste gebeurtenissen in javascript behoren:

De gebruiker klikt op de linkermuisknop of tikt met zijn vinger op een 'touch'scherm.	onClick()
De gebruiker dubbelklikt.	onDblClick()
De gebruiker beweegt met zijn muis of vinger over element op de pagina.	onMouseOver()
De gebruiker beweegt zijn muis of vinger van een element af.	onMouseOut()
De pagina is volledig geladen.	window.onblur() of in jQuery $ (document).ready()

In jQuery definieer je die 'events' vaak op een ietwat andere manier dan in standaard javascript.

Selecteer het element en koppel er een 'onClick'-event aan.	$(...).click();
Selecteer het element en koppel er een 'onMouseOver'-event aan.	$(...).mouseover();
Selecteer het element en koppel er een 'onMouseOut'-event aan.	$(...).mouseout();

In ons voorbeeld zou het er zo kunnen uitzien:

```
<script>
	$(document).ready(
		function(){
			$(“#BtnLeesArtikel1”).click();
			$(“#BtnLeesArtikel2”).mouseover();
		}
	);
</script>
```

STAP 5 : motoren starten bij een gebeurtenis (en ook STAP 6: het eindresultaat)

Uiteraard moet er ook nog iets gebeuren. Je moet nu nog een verbinding leggen tussen een bepaalde actie of “event” en een motor. M.a.w. “start een motor bij elk event”.

Een motor heet in javascript een “function” of een “method”.

Stap 1: het overzicht bewaren:

```
$(“#BtnLeesArtikel1”).click(

);
```

Stap 2: de functie toevoegen:

```
$(“#BtnLeesArtikel1”).click(
        function(){}
);
```

Stap 3: het overzicht bewaren:

```
$(“#BtnLeesArtikel1”).click(
        function(){

        }
);
```

Stap 3: het element selecteren waarop je de motor wil uitvoeren:

```
$(“#BtnLeesArtikel1”).click(
        function(){
                $(“#header”);
        }
);
```

Stap 4: een motor (methode) starten op het geselecteerde element:

```
$(“#BtnLeesArtikel1”).click(
        function(){
                $(“#header”).fadeOut();
        }
);
```

Welke methodes/motoren kan je standaard gebruiken?

1. HTML wijzigen

Je kan deze voorbeelden uittesten op http://www.computerkit.be/html5/index3.html

Net zoals in gewoon javascript is het heel eenvoudig om de HTML-inhoud van een bepaald element te wijzigen. Ook hier moet je eerste het element selecteren om er vervolgens de HTML-code van te wijzigen.

In javascript doen we dit als volgt:
document.getElementById("idVanGekozenElement").innerHTML= "nieuwe inhoud";

In jQuery kan je zoals gezien veel eenvoudiger een element selecteren:

$("#idVanGekozenElement").html("nieuwe inhoud");

In het gekozen voorbeeld wijzigen we de tekst van het element na het klikken op een knop. Na de klik verschijnt een nieuwe tekst en wordt de knop onzichtbaar:

```
<p id="veranderHTML">Dit is de originele tekst.</p>
 <button id="wijzigHTML">Verander de inhoud</button>
 <script>
  $(document).ready(function(){
   $("#wijzigHTML").click(function(){
    $("#veranderHTML").html("Dit is de nieuwe tekst die na de klik verschijnt.");
    $(this).hide();
   });
  });
 </script>
```

2. CSS wijzigen

Met jQuery kan je de CSS-eigenschappen van een object aanpassen. Dit is natuurlijk enkel zinvol als er zich eerst een 'gebeurtenis' voordoet, bijvoorbeeld als de gebruiker met de muis op een bepaald element klikt.

Een onderdeel van het HTML-bestand:

```
<button id="VeranderCSSRood">Rood</button>
<button id="VeranderCSSGroen">Groen</button>
<button id="VeranderCSSBlauw">Blauw</button>
<div id=”vierkant”></div>
```

Het onderdeel uit het CSS-bestand:

```
#vierkant{
        height:50px;
        width:50px;
        background-color:yellow;
}
```

Het script:

```
 <script>
 $(document).ready(function(){
   $("#VeranderCSSRood").click(function(){
      $("#vierkant").css("background-color", "red");
   });
   $("#VeranderCSSGroen").click(function(){
      $("#vierkant").css("background-color", "green");
   });
   $("#VeranderCSSBlauw").click(function(){
      $("#vierkant").css("background-color", "blue");
   });
 });
 </script>
```

Het vorige voorbeeld, veel korter, maar “level 2” in het schrijven van jQuery:

Het is ook mogelijk om ze de code uit het laatste voorbeeld verkort te schrijven wanneer we aan de buttons een CSS-klasse en een id geven.

Een onderdeel van het HTML-bestand:

```
<button class="veranderkleur" id="red">Rood</button>
<button class="veranderkleur" id="green">Groen</button>
<button class="veranderkleur" id="blue">Blauw</button>
<div id="vierkant"></div>
```

Het onderdeel uit het CSS-bestand:

```
#vierkant{
        height:50px;
        width:50px;
        background-color:yellow;
}
```

Het script:

Wanneer de gebruiker op een knop klikt, leest het script de “kleur” door te kijken naar de id van de knop. Elke knop heeft namelijk als “id” de naam van de gewenste kleur gekregen (vb. <button class="veranderkleur" id="red">Rood</button>).

Die kleur kan je in jQuery opvragen met $(this).attr("id");
$(this); betekent: neem *deze* knop die ik geselecteerd heb.

```
<script>
  $(document).ready(function(){
    $(".veranderkleur").click(function(){
        $("#vierkant").css("background-color", $(this).attr("id"));
    });
  });
</script>
```

3. Animaties bouwen

Zoals je wellicht al verwacht zitten in jQuery nog een aantal andere ingebouwde methodes om te animeren. Zo kan je .fadeOut() ook vervangen door andere methodes zoals bijvoorbeeld .slideDown(), .slideUp(), of .fadeIn().

Je kan methodes ook achter elkaar koppelen. Let echter op: een keten betekent niet dat alle methodes achter elkaar worden uitgevoerd. Veel 'methodes' worden zelfs tegelijkertijd uitgevoerd als je ze in een keten plaatst. In het voorbeeld hieronder wachten we daarom na elke methode 2 seconden:

```
<img src="examples/ballon.png" id="ballon3"  style="position:relative"/>
<script>
$(document).ready(function(){
  $("#ballon3").slideUp().delay(2000).slideDown().delay(2000).fadeOut().delay(2000).fadeIn();
});
</script>
```

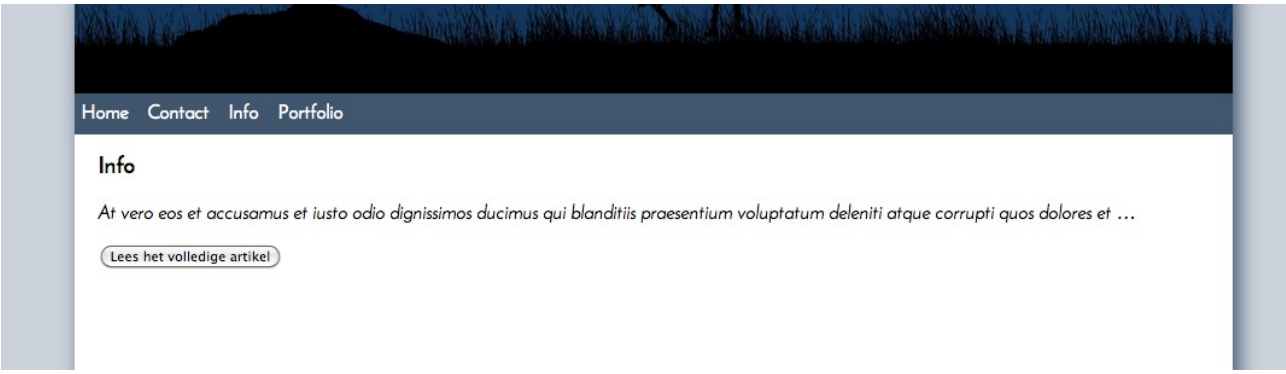
jQuery beschikt over het veel krachtiger .animate(). Tussen de haakjes, vertel je hoe jQuery het element moet animeren. Je bepaalt hier welke CSS-eigenschappen moeten geanimeerd worden én hoeveel tijd de animatie daarover moet doen.

In het onderstaande codevoorbeeld wordt het element ballon gedurende 5 seconden (5000 milliseconden) geanimeerd. Op het einde van de animatie zal het element 50 pixels meer naar rechts zijn geschoven. Het element 'springt' niet in één keer naar zijn nieuwe positie, maar je ziet het zachtjes schuiven. Niet alleen de positie aan de linkerkant, maar ook zijn graad van transparantie wordt geanimeerd.

```
<img src="images/ballon.png" id="ballon" style="position:relative"/>
<script>
$(document).ready(function(){
  $('#ballon').animate({
    opacity: 0.25,
    left: '+=50'
  },5000);
});
</script>
```

Een praktisch voorbeeld

We maken een script dat een bepaald onderdeel van de pagina eerst verbergt en bij een klik op een knop weer toont. Lijkt misschien gek, maar je ziet het vaak op krantenpagina's: enkel de inleiding van een artikel wordt getoond. Als de gebruiker op de knop “Lees het volledige artikel” klikt, wordt de rest van het artikel getoond.



Het onderdeel van de wegpagina:

De code hieronder bevat

- een titel (<h1>),
- een alinea (<p>) met de klasse “description” (<p class=description”>),
- een knop (<button>) met id “leesmeer” (<button id=”leesmeer”>),
- een div ( ) met id “verberg” ( ) die op zijn beurt dan weer een afbeelding (<img>) en een alinea (<p>) bevat.

```
<h1>Info</h1>

<p class="description">At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et …</p>

<button id="leesmeer">Lees het volledige artikel</button>

<div id="verberg">

        <img src="images/temp.jpg" class="rechts"/>

        <p>At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. </p>

</div>
```

Het script-blok

Het script verbergt eerst “automatisch” bij het laden van de pagina het onderdeel . Wanneer de gebruiker op de knop <button id="leesmeer">Lees het volledige artikel</button> klikt, verschijnt   weer in beeld. Op dat moment wordt ook de knop zelf verborgen.

```
<script>
	$(document).ready(function(){
		$("#verberg").hide();
		$("#leesmeer").click(function(){
			$(this).hide();
			$("#verberg").slideDown("slow");
		});
	});
</script>
```