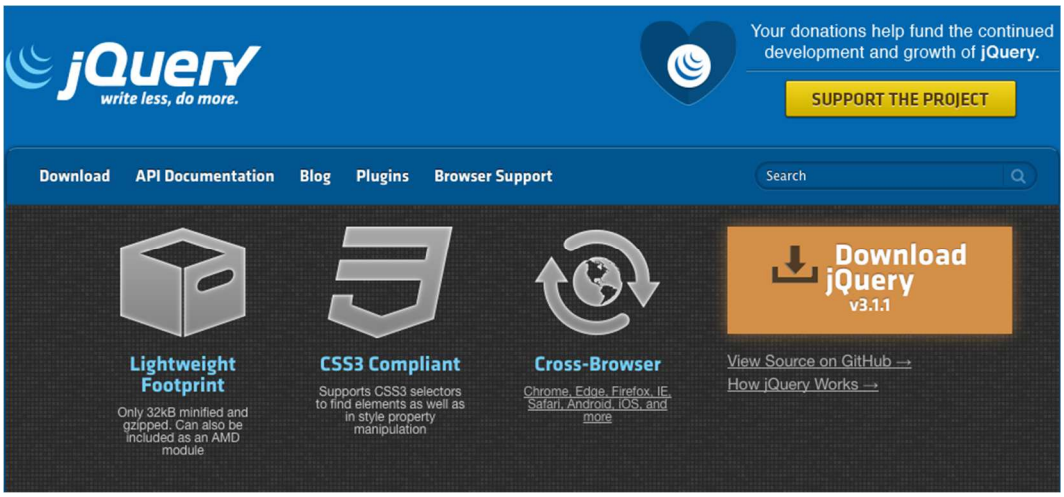
FRONTEND 2 - JQUERY en JQUERY UI

1. JQUERY

JQUERY is een JavaScript library die een hele reeks animaties, event handlers binnen HTML veel simpeler maakt over verschillende browsers heen. JQUERY leren doe je het best met voorbeelden van zijn mogelijkheden. In deze cursus zie je deze mogelijkheden en worden stap per stap besproken. Documentatie: api.jquery.com



1.1. READY EVENT

Het ready event zorgt ervoor dat alles binnen de DOM kan worden geladen. Alle methodes en properties die we aanspreken binnen het ready event zijn zo bereikbaar.

```
$(document).ready(function(){

});
```

1.2. JQOEF1: HIDE EVENT

Het hide event dient ervoor om tags op het scherm te laten verdwijnen.

1. Maak een map JQUERY aan.
2. Maak in de map JQUERY een map JQOEF1 aan.
3. Maak in de map JQOEF1 2 mappen aan: CSS en JS.
4. Maak een bestand style.css aan in de map CSS.
5. Maak een bestand index.js aan in de map JS.
6. Maak een bestand index.html aan in de map JQOEF1.
7. Download jquery.min.js en kopieer dit bestand in de map JS.

8. Pas als volgt aan in index.html:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/style.css">
    <title>JQUERY LEREN</title>
</head>
<body>
    <h1>JQUERY LEREN - HIDE EVENT</h1>
    <button>Klik op mij</button>
    <div class="box">
        <h3>Hallo 1!</h3>
    </div>
<script src ="js/jquery.min.js"></script>
<script src ="js/index.js"></script>
</body>
</html>
```

We voorzien een div tag met als class naam **.box**. Daarin staat er een h3 tag met tekst. Halo1!.

9. Pas als volgt aan in style.css:
We zullen de **.box** tag als volgt stylen:

```
.box{
    width:200px;
    border:1px solid #000;
    padding:15px;
    text-align:center;
    margin:0 auto;
}
```

9. Pas als volgt aan in index.js:
Wanneer we op de button klikken wordt de class **.box** aangesproken d.m.v. het hide event. Dit event zorgt ervoor dat de volledige div verdwijnt. Probeer maar uit!

```
//box verbergen : hide() methode
$(document).ready(function(){
    $('button').click(function(){
    $('.box').hide();
    });
});
```

1.3. JQOEF2: TOGGLE EVENT

Het toggle event heeft een dubbele functie in tegenstelling tot het hide event. Het toggle event bezit nl. 2 statussen nl. het tonen of het laten verdwijnen van een element.

1. Kopieer de vorige oefening en pas index.js als volgt aan:

```
//box toggelen
$(document).ready(function(){
    $('button').click(function(){
    $('.box').toggle();
    });
});
```

1.4. JQOEF3: HIDE EVENT MET SELECTOREN

Het hide event hebben we reeds besproken. We kunnen het hide event echter toepassen op een selector naar keuze. De selectoren die we kunnen gebruiken zijn o.a. first, last, odd en even. Probeer ze allemaal maar uit!

1. Kopieer de vorige oefening en pas index.html als volgt aan:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/style.css">
    <title>JQUERY LEREN</title>
</head>
<body>
    <button>Klik op mij</button>
    <div class="box">
        <h3>Hallo 1!</h3>
    </div>
    <div class="box">
        <h3>Hallo 2!</h3>
    </div>
    <div class="box">
        <h3>Hallo 3!</h3>
    </div>
    <div class="box">
        <h3>Hallo 4!</h3>
    </div>
<script src ="js/jquery.min.js"></script>
<script src ="js/index.js"></script>
</body>
</html>
```

2. Pas ook index.js aan:

```
//verberg met selectoren
//first, last.
//odd, even = oneven en even
$(document).ready(function(){
    $('button').click(function(){
    $('.box:first').hide();
    });
});
```

1.5. JQOEF4: TEXT OF HTML EVENT

Het HTML event dient om html EN hun tags toe te voegen vanuit javascript (jquery).

1. Kopieer de vorige oefening en pas index.js als volgt aan:
Probeer eerst het TEXT event en daarna het HTML event.

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box').html("<h1>Deze tekst komt van jquery</h1>");
    });
});
```

1.6. JQOEF5: ANIMATIE MET TIMING

We gaan deze animatie zonder chaining schrijven en met chaining. Tot nu toe hebben we onze oefeningen allemaal zonder chaining geschreven. JQUERY is echter in staat om alle events d.m.v. een punt aan elkaar te plakken. In bijvoorbeeld PHP spreken we van concatenating, hier spreken we van CHAINING.

1. Kopieer de vorige oefening en pas index.js als volgt aan:

ZONDER CHAINING:

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box:first').css("color","red");
    $('.box:first').css("border","2px solid red");
    $('.box:first').hide(2000);
    });
});
```

MET CHAINING:

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box:first').css("color","red").css("border","2px solid red").hide(2000);
    });
});
```

1.7. JQOEF6: APPEND, AFTER en BEFORE EVENT

Je hebt de mogelijkheid om tekst toe te voegen maar dit keer bepalen wij de locatie waar we deze tekst laten verschijnen. We spreken een div tag aan en kiezen dan het juiste event op de locatie te bepalen. Deze events zijn: append, after en before.

1. Kopieer de vorige oefening en pas index.js als volgt aan:

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box').append("<h3>Dit is jquery tekst aanvulling in de box<h3>");
    $('.box').after("<h3>Dit is jquery tekst achter de box<h3>");
    $('.box').before("<h3>Dit is jquery tekst voor de box<h3>");
    });
});
```

1.8. JQOEF7: PROPERTIES

In de voorgaande oefeningen hebben we het reeds gehad over selectoren om 1 bepaald element aan te spreken. Naast deze selectoren kunnen we ook de properties aanspreken en eigenlijk de css wijzigen vertrekkende vanuit JQUERY.

1. Kopieer de vorige oefening en pas index.js als volgt aan:
Hier zullen we één property toepassen op onze box. we zullen namelijk de kleur veranderen in rood.

EEN PROPERTY:

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box:first').css("color","red")

    });
});
```

MEERDERE PROPERTIES:

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box:first').css({"color":"red","border":"2px solid blue"});

    });
});
```

1.9. JQOEF8: CLASSES TOEVOEGEN

Vanuit JQUERY hebben we een eenvoudige manier om classes toe te voegen aan html elementen. Dit event is het **addClass** event.

1. Kopieer de vorige oefening en pas style.css als volgt aan:
We maken eerst een nieuwe class aan in ons style.css bestand.

```
.box{
    width:200px;
    border:1px solid #000;
    padding:15px;
    text-align:center;
    margin:0 auto;
    transition: all 0.6s linear;
}
.color{
    background-color: purple;
    color:white;
}
```

2. pas nu het index.js bestand aan:
Zoals je kan zien voegen we nu deze class toe aan het box element. We dienen dus dit niet hardcore toe te voegen in ons index.html bestand.

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box').addClass('color');

    });
});
```

Wanneer we addClass zouden vervangen door het toggleClass event dan switchen we de css styling dynamisch aan en uit.

```
$(document).ready(function(){
    $('button').click(function(){
    $('.box').toggleClass('color');

    });
});
```

1.10. JQOEF9: ATTR PROPERTY

1. Kopieer de vorige oefening en pas index.html als volgt aan:
Voeg ook 2 afbeeldingen toe.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/style.css">
    <title>JQUERY LEREN</title>
</head>
<body>
    <button>Klik op mij</button>
    <div class="box">
        <h3>Hallo 1!</h3>
        <img src="Syntrawest_top.png" width="100%" height="100%" alt="">
    </div>

<script src ="js/jquery.min.js"></script>
<script src ="js/index.js"></script>
</body>
</html>
```

Pas index.js als volgt aan:

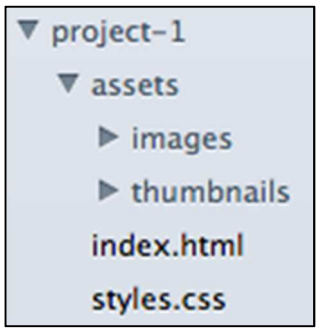
```
$(document).ready(function(){
    $('button').click(function(){
    $('img').attr("src","sbm.jpeg");

    });
});
```

2. EERSTE PROJECT : JQUERY LIGHTBOX

In ons eerste project gaan we de gekende lightbox voor foto's namaken. Er bestaat namelijk een lightbox library die opgebouwd is uit html, css, javascript en jawel JQUERY. Dergelijke libraries worden third-party libraries genoemd.

1. Maak de volgende structuur aan:



2. Zoek op google afbeeldingen 6 foto's van dezelfde grootte en plak ze in de images en thumbnails map.

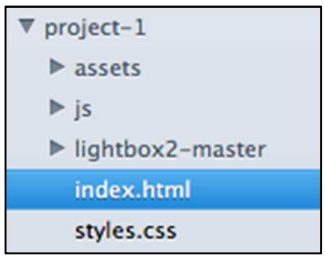
3. Maak je index.html bestand als volgt op en zorg ook hier dat je verwijst naar jquery.min.js. Je dient dus deze folder (js) in je project te voorzien.

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width,
        initial-scale=1">
        <title>Lightbox implementatie</title>
        <!-- CSS -->
        <link href="styles.css" rel="stylesheet">
    </head>
    <body>
        <div class="container">
            <header>
                <span>-  Vakantie  -</span>
                <h1>Onze foto's</h1>
            </header>
            <!-- Thumbnail Images -->
            <img src="assets/thumbnails/thumb1.jpg" alt=""/>
            <img src="assets/thumbnails/thumb2.jpg" alt=""/>
            <img src="assets/thumbnails/thumb3.jpg" alt=""/>
            <img src="assets/thumbnails/thumb4.jpg" alt=""/>
            <img src="assets/thumbnails/thumb5.jpg" alt=""/>
            <img src="assets/thumbnails/thumb6.jpg" alt=""/>
        </div>
        <!-- JavaScript -->

        <script src="js/jquery.min.js"></script>
    </body>
</html>
```

4. Ga naar de volgende shortcut link. We gaan de lightbox2 library downloaden:
http://lokeshdhakar.com/projects/lightbox2/

Download en unzip deze library en plak hem in je project folder:



Je kan op de pagina van de lightbox2 ontwerper de stappen terugvinden om dit toe te voegen aan je project.

Lightbox heeft dus zijn eigen javascript bestand. Dit bestand heeft de jquery library ook nodig. Zorg er dus steeds voor dat de jquery.min.js VOORDIEN is geladen! Merk op dat in bovenstaande links het juiste path moet staan!

Ga naar de index.html pagina.
Eerst voeg je de css toe:

```
<link href="styles.css" rel="stylesheet">
<link href="lightbox2-master/dist/css/lightbox.css" rel="stylesheet">
```

Daarna voeg je het script toe:

```
<script src="js/jquery.min.js"></script>
<script src="lightbox2-master/dist/js/lightbox.js"></script>
```

5. Vervolgens dienen we de html te initialiseren zoals lightbox dit voorzien heeft.

Add a `data-lightbox` attribute to any image link to enable Lightbox. For the value of the attribute, use a unique name for each image. For example:

```
<a href="images/image-1.jpg" data-lightbox="image-1" data-title="My caption">Image #1</a>
```

Optional: Add a `data-title` attribute if you want to show a caption.

If you have a group of related images that you would like to combine into a set, use the same `data-lightbox` attribute value for all of the images. For example:

```
<a href="images/image-2.jpg" data-lightbox="roadtrip">Image #2</a>
<a href="images/image-3.jpg" data-lightbox="roadtrip">Image #3</a>
<a href="images/image-4.jpg" data-lightbox="roadtrip">Image #4</a>
```

Pas je index.html pagina als volgt aan.

```
<a href="assets/images/image1.jpg" data-lightbox="image1" data-title="My caption">
    <img src="assets/thumbnails/thumb1.jpg" alt=""/>
</a>
<a href="assets/images/image2.jpg" data-lightbox="image2" data-title="My caption">
    <img src="assets/thumbnails/thumb2.jpg" alt=""/>
</a>
<a href="assets/images/image3.jpg" data-lightbox="image3" data-title="My caption">
    <img src="assets/thumbnails/thumb3.jpg" alt=""/>
</a>
<a href="assets/images/image4.jpg" data-lightbox="image4" data-title="My caption">
    <img src="assets/thumbnails/thumb4.jpg" alt=""/>
</a>
<a href="assets/images/image5.jpg" data-lightbox="image5" data-title="My caption">
    <img src="assets/thumbnails/thumb5.jpg" alt=""/>
</a>
<a href="assets/images/image6.jpg" data-lightbox="image6" data-title="My caption">
    <img src="assets/thumbnails/thumb6.jpg" alt=""/>
</a>
```

Wanneer je bovenstaande nu in je browser bekijkt dan zal je iedere foto afzonderlijke kunnen laden, maar kunnen we niet van de ene naar de andere foto in de lightbox doorklikken. Daarvoor hebben we de **data-lightbox** property nodig. Wanneer je in tegenstelling tot bovenstaande schermweergave deze property overal dezelfde naam geeft, dan worden al deze afbeeldingen door lightbox als groep ervaren en kun je in de lightbox doorklikken naar de volgende afbeelding.
Pas data-lightbox aan als volgt:
data-lightbox = "vakantie"

6.
Extra gedragingen: er zijn enkele extra eigenschappen toegevoegd om te lightbox aan te passen en te stylen volgens uw wensen.

Wanneer je op de pagina van de developer kijkt dan zie je de volgende script tag staan die je kan toevoegen aan je pagina waar lightbox wordt opgeroepen. Je kan dus nog voor extra functionaliteit zorgen.

```
<script>
    lightbox.option({
      'resizeDuration': 200,
      'wrapAround': true
    })
</script>
```

Voeg bovenstaande script toe aan je index.html pagina en pas als volgt aan:
We positioneren de lightbox op 200px van de top van het scherm ipv 50 die als standaard wordt meegegeven.

```
    <script>
        lightbox.option({
      'positionFromTop': 200

    })
    </script>
</body>
```

Bijkomende optiemogelijkheden die je kan uitproberen zijn:

Option	Default	Description
alwaysShowNavOnTouchDevices	false	If true, the left and right navigation arrows which appear on mouse hover when viewing image sets will always be visible on devices which support touch.
albumLabel	"Image %1 of %2"	The text displayed below the caption when viewing an image set. The default text shows the current image number and the total number of images in the set.
disableScrolling	false	If true, prevent the page from scrolling while Lightbox is open. This works by settings overflow hidden on the body.
fadeDuration	600	The time it takes for the Lightbox container and overlay to fade in and out, in milliseconds.
fitImagesInViewport	true	If true, resize images that would extend outside of the viewport so they fit neatly inside of it. This saves the user from having to scroll to see the entire image.
imageFadeDuration	600	The time it takes for the image to fade in once loaded, in milliseconds.
maxWidth		If set, the image height will be limited to this number, in pixels.
maxHeight		If set, the image width will be limited to this number, in pixels.
positionFromTop	50	The distance from top of viewport that the Lightbox container will appear, in pixels.
resizeDuration	700	The time it takes for the Lightbox container to animate its width and height when transition between different size images, in milliseconds.
showImageNumberLabel	true	If false, the text indicating the current image number and the total number of images in set (Ex. "image 2 of 4") will be hidden.
wrapAround	false	If true, when a user reaches the last image in a set, the right navigation arrow will appear and they will be to continue moving forward which will take them back to the first image in the set.

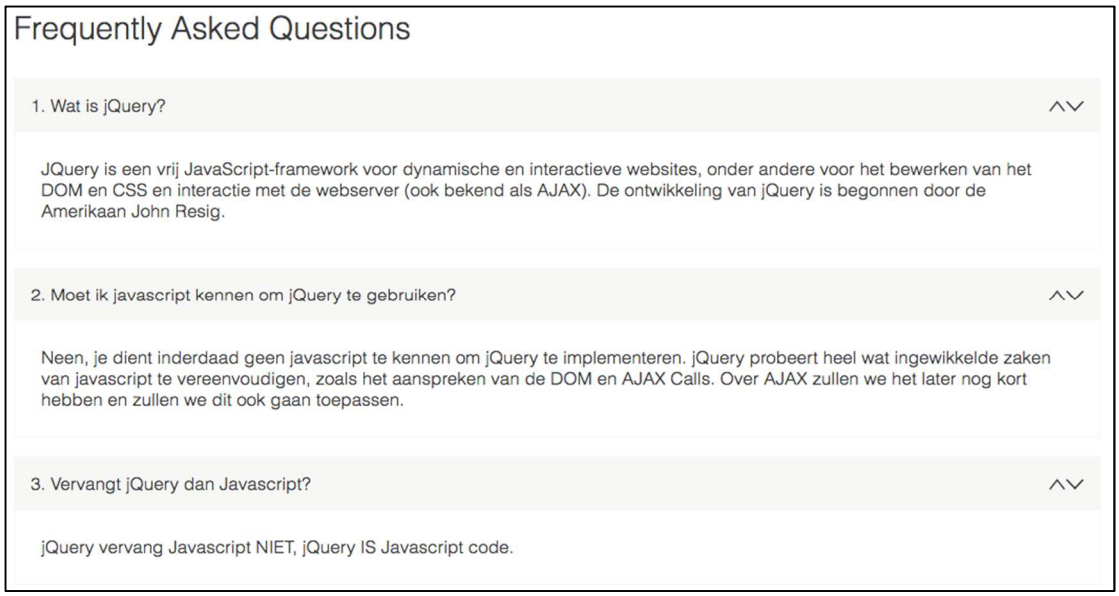
3. TWEEDE PROJECT : JQUERY FAQ

1.
Open de index.html pagina van het project.

Onderstaand schreenshot zou je moeten zien staan.

Bovenstaande pagina is nog steeds statisch. Wij zullen deze dynamisch maken d.m.v. het gebruik van jQuery.
Zoals je kan zien worden de antwoorden niet getoond, dit komt omdat we deze in ons css bestand op **display:none** hebben geplaatst. De klasse die we hiervoor gebruiken binnen onze html tags is **.collapse**.

Schakel even de collapse uit in je css, zodat je je volledige pagina ziet staan.



Nu zien we perfect welke elementen we dienen aan te spreken.

2.
Maak nu je index.js bestand aan naast je index.html bestand. Hier zullen we onze eigen code schrijven.

Voeg ook het index.js als script source toe aan je index.html bestand. Foutje in onderstaand screenshot, wijzig naar $("#arrow1-down, arrow1-up").

```
$("#q1").click(function(){
    $("#a1").fadeToggle('slow');
    $("#arrow1-down", "#arrow1-up").toggleClass('collapse');
});
$("#q2").click(function(){
    $("#a2").fadeToggle('slow');
    $("#arrow1-down", "#arrow1-up").toggleClass('collapse');

});
$("#q3").click(function(){
    $("#a3").fadeToggle('slow');
    $("#arrow1-down", "#arrow1-up").toggleClass('collapse');
});
```

Bovenstaande is perfecte code, maar dit kunnen we vereenvoudigen. In javascript hebben we het reeds gehad over het DOM (Document Object Model). Dit dien je zeer goed te begrijpen om jquery ten volle te kunnen benutten, m.a.w. op de api van jquery te gebruiken. De documentatie voor de API van jquery kan je terugvinden op **api.jquery.com**.

3.
Wanneer we op gelijk welke element binnen de DOM aanspreken, m.a.w. binnen onze html tags, dan stelt jquery dit gelijk met het keyword **this**.

Wanneer we in onze index.html kijken, dan zien we dat iedere vraag ook dezelfde classe heeft.

Pas je code als volgt aan:

```
$(".vraag").click(function(){
    console.log($(this));
});
```

Wanneer we volgens bovenstaande code op een vraag klikken dan zal de console ons tonen op WELK element we geklikt hebben.

```
▶ [div#q1.vraag, context: div#q1.vraag]
▶ [div#q2.vraag, context: div#q2.vraag]
▶ [div#q3.vraag, context: div#q3.vraag]
```

4.
De DOM is eigenlijk een boomstructuur. Deze boomstructuur kunnen we gemakkelijk gaan benaderen in jQuery d.m.v. **Traversing**. Al de mogelijkheden van traversing kun je op api.jquery.com terugvinden. De 3 grote onderdelen van traversing zijn: parents, children en siblings (tweelingen = zelfde niveau).

Wanneer we ons html bestand van nabij bekijken, dan zien we dat de vragen en antwoorden op hetzelfde niveau zitten en dus eigenlijk tweelingen zijn.

Wanneer ik dus klik op een vraag dan dien ik mijn sibling of volgende element terug te geven.

We kunnen dus gewoon gebruik maken van het volgende element, nl. next(). Sibling kan je natuurlijk ook gebruiken.

Dus: ik klik op een vraag (this) en geef het volgende html element terug (next).

```
$(".vraag").click(function(){
    console.log($(this).next());
});
```

Wanneer je dit test in je console dan zie je het volgende resultaat:

```
▶ [div#a1.antwoord.collapse, prevObject: m.fn.init[1], context: div#q1.vraag]
```

Bovenstaand geeft je dus wel degelijk het **next** html element en dit is net wat we willen tonen! Nu we weten dat dit werkt kunnen we ons code als volgt aanpassen met chaining.

```
$(".vraag").click(function(){
$(this).next().fadeToggle("fast");
});
```

5.
Op dezelfde manier kan je omgaan met de pijltjes die je dient weer te geven. Probeer dit als oefening maar eens uit.

```
$(".vraag").click(function(){
$(this).next().fadeToggle("fast");
$(this).children().toggleClass('collapse');
});
```

4. DERDE PROJECT : TODO LIJST

Voor ons derde project gaan we een todo lijst maken.

1.
Open de index.html pagina van het project. Onderstaande lay-out is reeds te zien maar is nog niet functioneel.

2.
Aangezien we met forms werken kunnen we in api.jquery.com de rubriek forms raadplegen. Onderaan zien we de standaard functie submit() staan die we gebruiken in onze submit knop van een formulier.

```
$(document).ready(function () {
    $('add-items').submit(function(){
        console.log('test of de functie submit werkt.');
    });
});
```

Wanneer je nu een taak intikt en je klikt op taak toevoegen dan zal je zeer kort zien dat de submit functie zijn werk probeert uit te voeren. In het geval van een form zal de submit button bijvoorbeeld proberen alle data weg te schrijven naar een database.

Aan de hand van bovenstaand voorbeeld kunnen we de **event.preventDefault()** methode nu klaar en duidelijk uitleggen. De submit() methode voert namelijk zijn standaard gedraging uit! Dit willen we onderbreken en daarvoor dient de preventDefault() methode. We geven aan de functie de parameter event mee en zorgen ervoor dat via even.preventDefault() de standaard functionaliteit van submit wordt onderbroken.

```
$(document).ready(function () {
    $('.add-items').submit(function(event){
        event.preventDefault();
        console.log('test of de functie submit werkt.');
    });
});
```

3.
Dit werkt nu. Je kan nu de controle van de console.log lijn wissen.
De volgende stap is nu om de input van de gebruiker op te vangen.
We weten dat de inputbox als id **#todo-list-item** heet. Wanneer de gebruiker iets in de inputbox intikt dan gebruiken we de **val()** methode om de tekst op te vangen in een variabele. Deze variabele **item** heeft dus een uit te voeren taak vast.
Vervolgens gaan we deze taak afdrukken op de hiervoor voorziene locatie in ons html document. Deze locatie is een unordered list die we gaan vullen met list-items. De unordered list is gekend, nl. **#list-items.** Daarna gaan we telkens een toevoeging van iedere taak uitvoeren door de **append()** methode te gebruiken.
Als laatste gaan we dynamisch de ** list-item tags** gaan genereren met daartussen de variabele die de taak bevat.

```
$(document).ready(function () {
    $('.add-items').submit(function(event){
        event.preventDefault();
        var item = $('#todo-list-item').val();
        $('#list-items').append('<li>'+ item + '</li>');
    });
});
```

Probeer maar uit! Je kan nu je taken ingeven.

4.
We breiden dit uit met een checkbox en een verwijder knop.

```
$(document).ready(function () {
    $('.add-items').submit(function(event){
        event.preventDefault();
        var item = $('#todo-list-item').val();
        $('#list-items').append("<li><input type='checkbox'/>" + item + "<a class='remove'>x</li>");
    });
});
```

5.
Telkens er een taak wordt bijgevoegd blijft de input momenteel staan. De inputbox dient dus ook te worden leeg gemaakt.
Pas hiervoor onderstaande code aan:

```
$(document).ready(function () {
    $('.add-items').submit(function(event){
        event.preventDefault();
        var item = $('#todo-list-item').val();
        $('#list-items').append("<li><input type='checkbox'/>" + item + "<a class='remove'>x<hr></li>");
        $('#todo-list-item').val("");
    });
});
```

6.
Bovenstaande werkt perfect, maar wanneer we een leeg veld proberen toe te voegen dan werkt dit ook.

OEFENING: los dit op!
OPLOSSING:
if statement toevoegen met de test op item.

```
event.preventDefault();
var item = $('#todo-list-item').val();
if(item){
$('#list-items').append("<li><input class='checkbox' type='checkbox'/>" + item +
$('#todo-list-item').val("");
}
```

7.
We dienen ook nog te zorgen dat wanneer een taak is uitgevoerd en de checkbox wordt aangevinkt, de taak ook is afgewerkt. Daarvoor moeten we de class completed kunnen gebruiken uit ons css bestand.
In het vorige scherm mag je ook niet vergeten de class checkbox eerst toe te voegen.
Het on event van de DOM (document) heeft 3 parameters. De eerste is het change event, d.w.z. wanneer er maar iets veranderd aan een object dient function te worden uitgevoerd. Het object dat we controleren is de 2de parameter, nl. de classe .checkbox!
Aan de parent tag wordt dus het li element wordt de class completed toegevoegd en wordt de taak doorstreept

```
$(document).on("change", ".checkbox", function(){
    $(this).parent().toggleClass("completed");
});
```

8.
OEFENING: probeer nu de code te schrijven om de lijn te verwijderen.
OPLOSSING:

```
$(document).on("click", ".remove", function(){
    $(this).parent().remove();
});
```

9.
Wanneer we nu onze pagina verversen dan verliezen we al onze taken en dit is niet de bedoeling. Sinds HTML5 heeft de front end ook de mogelijkheid om de database van de browser te gaan gebruiken. Inderdaad een browser heeft een local database.

Open je inspect element en ga naar de **Resources** of de **Application** tab. Daar zie je een link **Local Storage**.

```
$(document).ready(function () {
    $("#list-items").html(localStorage.getItem("listItems"));

    $(".add-items").submit(function (event) {
        event.preventDefault();

        var item = $("#todo-list-item").val();

        if(item){
            $("#list-items").append("<li><input class='checkbox' type='c
            localStorage.setItem("listItems", $("#list-items").html());
```

We kunnen nu onze pagina verversen zonder dat we onze taken verliezen.

10.
Wanneer we nu een taak verwijderen, dan dient die ook uit de localStorage te verdwijnen. we kunnen dit doen door onze code te kopiëren en deze in onze remove function te plakken.

```
$(document).on("change", ".checkbox", function(){
    $(this).parent().toggleClass("completed");
    localStorage.setItem("listItems", $("#list-items").html());
});

$(document).on("click", ".remove", function(){
    $(this).parent().remove();
    localStorage.setItem("listItems", $("#list-items").html());
});
```

11.
We kunnen nu terug gegevens uit de localStorage verwijderen. Wanneer we echter ons venster refreshen dan zien we dat geschrapte taken ook mooi geschrapt blijven, maar dat ons vinkje niet meer aan staat van onze geschrapte taak.

OEFENING: LOS OP!
OPLOSSING:

```
$(document).on("change", ".checkbox", function(){
    if($(this).attr("checked")){
        $(this).removeAttr('checked');
    }else{
        $(this).attr("checked", "checked");
    }
```