Attach events

Er zijn verschillende mogelijkheden om events te attachen en te detachen.
We kunnen ze rechtstreeks attachen, of via een ander mechanisme.

Er zijn verschillende manieren

- click()
- bind() & unbind()
- live() & die()
- delegate() & undelegate()
- on() & off()

bind()

Elke keer dat we een event attachen, wordt dit in memory bijgehouden.
Om ervoor te zorgen dat we geen memoryleaks hebben en dat we niet teveel memory consumeren, kunnen we de events terug detachen.

De rechtstreekse method is een verkorte versie van de bind versie.
Onderstaande zijn exact twee dezelfde methods.

```
$('selector').click(function () {
        alert('clicked!!!');
});

$('selector').bind('click', function () {
    alert('clicked!!!');
});
```

We kunnen deze twee methods ook unbinden.

```
$('selector').unbind('click');
```

live()

In jQuery 1.3 hebben ze live geïmplementeerd, deze hebben ze ook in jQuery 1.4.3 ook terug verwijdert.
Voorbeelden:

```
$('selector' ).live( "click", function() {
    alert('clicked!!!');
});

$('selector').die('click');
```

delegate()

Vanaf jQuery 1.4 kunnen we delegate gebruiken.
Voorbeelden:

```
$(document).delegate('selector','click', function () {
    alert('clicked!!!');
});

$(document).undelegate('selector', 'click');
```

on()

Vanaf jQuery 1.7 werd de on() event geïntroduceerd.
Het wordt aangeraden om deze te gebruiken.
Deze vervangt de live() en delegate() events.
Dus vanaf werken we enkel nog met het on event.

Voorbeelden:

```
$( 'selector' ).on( 'click', function () {
        alert('clicked!!!');
});

$(document).on('click', 'selector', function () {
    alert('clicked!!!');
});
```

We kunnen ook gebruik maken van meerdere events in één handler.

```
$(document).on('mouseenter mouseleave', 'selector', function () {
        $(this).toggleClass('highlight');
});
```

We kunnen ook gebruik maken van de map() method, dit is eigenlijk een jSON object:

```
$('selector').on({
        mouseenter: function () {
                $(this).addClass('highlight');
        }, mouseleave: function () {
                $(this).removeClass('highlight');
        }
});
```

Waarom on() gebruiken ?

Er zijn twee redenen om het on() event te gebruiken.

1. Eén event attachen.

Stel je voor dat je een tabel hebt met 1000 rijen.
Op elke rij heb een click event gedefiniëerd.
Al gauw heb je dus 1000 click events voor één enkele tabel.
Neem nu dat je nog een hover event hebt voor elke rij, en je zit weer aan een duizend meer.

Als je volgend event gebruikt, heb je maar één click event.

```
$('table').on('click', 'tr', function () {
    //DOWORK
});
```

2. Events attachen, op elementen die on the fly zijn gemaakt.

Er zijn twee manieren om een event aan een element te attachen, dat dynamisch aangemaakt wordt.
Je kunt dit doen wanneer je het element aanmaakt.
We gaan een p-tag dynamisch aanmaken en attachen rechtstreeks een click event aan het element.
Dit is een methode hoe je een p-tag kan aanmaken.
Het nadeel is dat je zo 1000 events aanmaakt.

```
var $div = $('div');

for (x = 0; x < 1000; x++) {
    var $p = $('<p>').text('Ik ben een paragraaf on the fly').click(function () {
        $(this).css('color', 'red');
    });
    $div.append($p);
};
```

Een betere manier is om eerst een event handler te maken op de div-tag en daarna kan dit van toepassing zijn op elke p-tag dat je later nog zou aanmaken.

```
$('div').on('click', 'p', function () {
    $(this).css('color', 'red');
});

var $div = $('div');

for (x = 0; x < 1000; x++) {
    var $p = $('<p>').text('Ik ben een paragraaf on the fly');
    $div.append($p);
};
```

Als je de on event gebruikt moet je wel zien dat je een element neemt, dat geen performance issues geeft.
Je kan bijvoorbeeld ook het event aan het document attachen, maar dan zou het heel het document moeten scannen om je p-tag te vinden.

```
$(document).on('click', 'p', function () {
    $(this).css('color', 'red');
});
```

Bubbling

Hoe werkt nu dit on() event ?
Elk event occured op het laagste element.
Waarna het event naar boven bubbled naar de bovenliggende elementen, tot aan het document.
Voorbeeld als je nested elementen hebt.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Events</title>
        <script src="jquery-1.10.2.js"></script>
        <script src="Script.js"></script>

    </head>
    <body>
        <div>
            <table>
                <tbody>
                <tr>
                    <td><p>cell</p></td>
                </tr>
            </tbody>
            </table>
        </div>
    </body>
</html>
```

Als je op cell klikt, gaat het kijken eerst op de paragraaf, of er een event is.
Daarna gaat het één hoger, en zo bubbelt het naar boven.
De on vangt dit op op een niveau, hij gaat dan terug kijken welk child de target is, en daar het event op toepassen.

Je kunt de volgorde zien door volgend script toe te voegen.

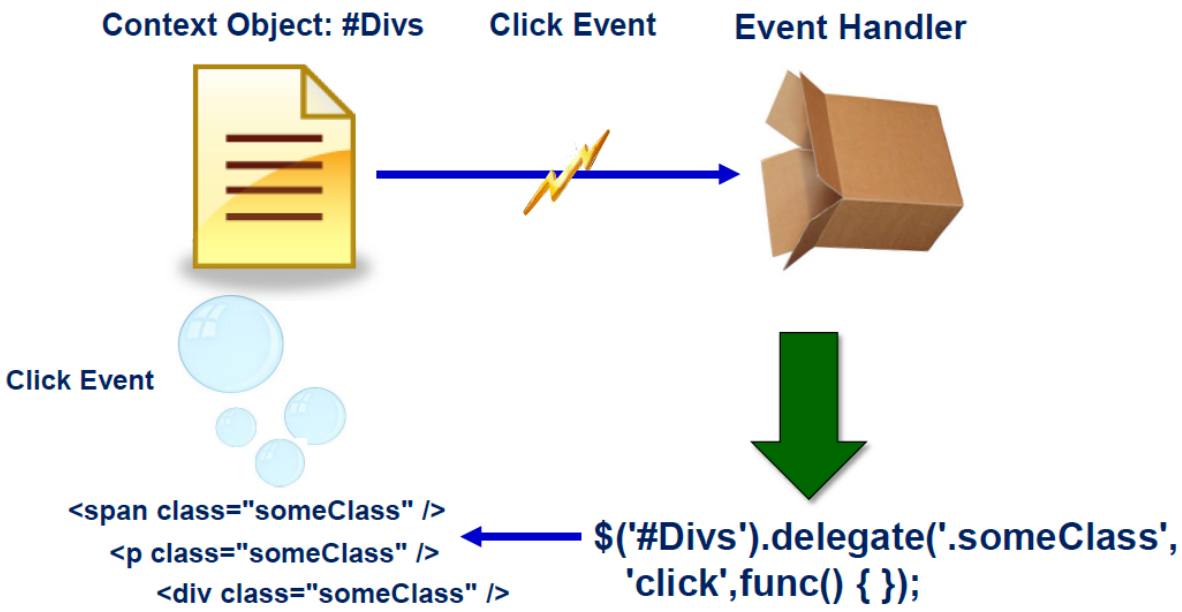
```
$(function () {
    $('div').click(function (e) {
        $(alert(e.currentTarget));
    });

    $('table').click(function (e) {
        $(alert(e.currentTarget));
    });

    $('tr').click(function (e) {
        $(alert(e.currentTarget));
    });

    $('td').click(function (e) {
        $(alert(e.currentTarget));
    });

    $('p').click(function (e) {
        $(alert(e.currentTarget));
    });
});
```


Context Object: #Divs
Click Event
Event Handler
Click Event

<p class="someClass" />

$('#Divs').delegate('.someClass',
'click',func() { });