

Korte Herhaling

- Framework 7 – Statische componenten
 - Button, Card, Chip, etc.
 - Voornamelijk HTML, user interface onderdelen
- Grid Systeem
- Kleuren
- Icons
- Dynamische componenten
 - Inleiding
 - Voornamelijk JavaScript, interactie onderdelen

Vragen?

Vandaag

- Dynamische componenten
 - JavaScript begrijpen
 - Stap voor stap opbouwen - Gauge
 - Voorbeelden: Picker, Dialog en meer.
- Ajax
 - Communicatie met webservers
 - (Formulieren posten)
- Case

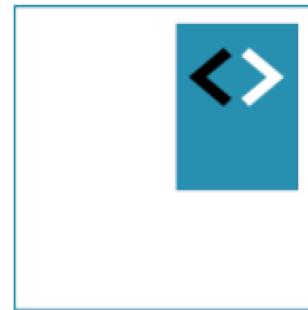


Ministerie van Defensie

Crossplatform Mobile Apps

Module 5 – Framework 7

Dynamische elementen



Peter Kassenaar – info@kassenaar.com

Dynamische componenten

- De pagina wordt zelf een 'component' met <template>
- <template> tags worden niet getoond, alleen via JavaScript
- De lay-out is nu anders:

```
<!-- dialog.html -->
<template>
  <div class="page" data-name="...">
    <div class="navbar">
      ...
    </div>
    <div class="page-content">
      ...
    </div>
  </div>
</template>
```

Dynamische componenten

- Dynamische componenten hebben altijd een <script> tag
- Componenten moeten minimal een JavaScript object retourneren
- De *inhoud* van het JavaScript object lees je in de documentatie

```
1 <template>
2 ...
3     <div class="page">
4         <div class="page-content">
5             Inhoud van de pagina/component
6         </div>
7     </div>
8 </template>
9 <script>
10    return {
11        // JavaScript-inhoud van de component
12    }
13 </script>
```

Routing aanpassen

- De routing is nu anders:

```
routes: [  
  ...  
  {  
    path: '/dialog/',  
    componentUrl: './pages/dialog.html'  
  }  
]
```



Wanneer moet je dit doen?

Als de pagina dynamische framework7-elementen bevat, zoals
Dialog, Gauge, Login Screen, Photo browser
enzovoort.

Dynamische componenten

- Hoe weet je of een Framework7-component een dynamische component is?
 - Lees de documentatie.
 - Er zit (volgens mij) niets anders op.
- Als er documentatie staat over Methods, Events, Properties, dan heb je te maken met een dynamische component

The screenshot shows the Framework7 documentation for the Popover component. On the left, there's a sidebar with various components listed: Dialog, Elevation, Floating Action Button / FAB, Form Data / Storage, Gauge, Grid / Layout Grid, Icons, Infinite Scroll, Inputs / Form Inputs, Lazy Load, Link, List View, List Index, Login Screen, Messagebar, Messages, Navbar, Notification, Page, Panel / Side Panels, Photo Browser, Picker, and Popover. The 'Popover' item is highlighted with a red arrow pointing to it from the bottom-left. The main content area has a dark header with '</body>' and a sub-header 'Popover App Methods'. It contains text about the Popover component being highly customizable and notes that it can contain anything, even another view with navigation. Below this, three methods are listed: `app.popover.create(parameters)`, `app.popover.destroy(el)`, and `app.popover.get(el)`. A red callout box on the right side of the page states 'Popover is een dynamische component'.

Popover is a highly customizable element and you can put anything inside, even another view with navigation.

Popover App Methods

Let's look at related App methods to work with Popover:

- `app.popover.create(parameters)` - create Popover instance
 - `parameters` - object. Object with popover parameters

Method returns created Popover's instance
- `app.popover.destroy(el)` - destroy Popover instance
 - `el` - HTMLElement or string (with CSS Selector) or object. Popover element or Popover instance to destroy.
- `app.popover.get(el)` - get Popover instance by HTML element
 - `el` - HTMLElement or string (with CSS Selector). Popover element.

Method returns Popover's instance

Popover is een dynamische component

De voorbeelden hierna zijn *generieke* voorbeelden

Zorg er voor dat je de algemene stappen kent (template, script, object uitbreiden). Dan kun je ook relatief snel met de andere dynamische componenten aan de slag.

Deze werken op vergelijkbare wijze.

Een component maken

1. Maak een nieuwe pagina op de gebruikelijke manier
2. Denk er aan de routes in `app.js` uit te breiden
3. Plaats de `<template>` tags in de pagina
4. Vul de pagina met je gewenste inhoud
5. Plaats de `<script>` tags
6. Vul de script-tags volgens de documentatie
7. Voorbeeld Gauge en range-slider!



1. Nieuwe pagina maken

```
1 <template>
2 <div class="page">
3
4     <div class="navbar">...
14    </div>
15    <div class="page-content">
16        ... hier komt de inhoud ...
17    </div>
18 </div>
19
20 </template>
```

2. HTML toevoegen

- Statische gauge toevoegen, kopieren vanuit HTML-documentatie

```
</div>
<div class="page-content block">
    <h1>Mijn salarismeter</h1>
    <div class="gauge salaris-gauge"></div>
</div>
```

3. Script Toevoegen

- Object retourneren, statische code voor gauge invoegen in on:pageInit hook

```
<!-- Hier komt het script -->
<script>
    return {}
</script>
```

Page lifecycle hooks

- On: pageInit

```
return {
  on: {
    pageInit: function () {
      console.log('De gauge wordt geinitialiseerd');
      var gauge = app.gauge.create({
        // .. hier komt straks nuttige code.
      });
    }
}
```

Standaard waarden opgeven als parameters

```
on: {
    pageInit: function () {
        console.log('De gauge wordt geinitialiseerd');
        // Stop de gauge in een variabele,
        // zodat je hem later kunt updaten...
        var gauge = app.gauge.create({
            el: '.salaris-gauge',
            type: 'circle',
            value: 0.3,
            size: 250,
            borderColor: '#2196f3',
            borderBgColor: 'pink',
            borderWidth: 20,
            valueText: '30%',
            valueFontSize: 41,
            valueTextColor: '#2196f3',
            labelText: 'Mijn salaris',
        });
    }
}
```

Slider toevoegen

- HTML voor Range slider toevoegen

```
<!-- range slider -->
<div id="slider" class="range-slider range-slider-init">
|   <input type="range" min="0" max="100" step="1" value="50">
</div>
</div>
```

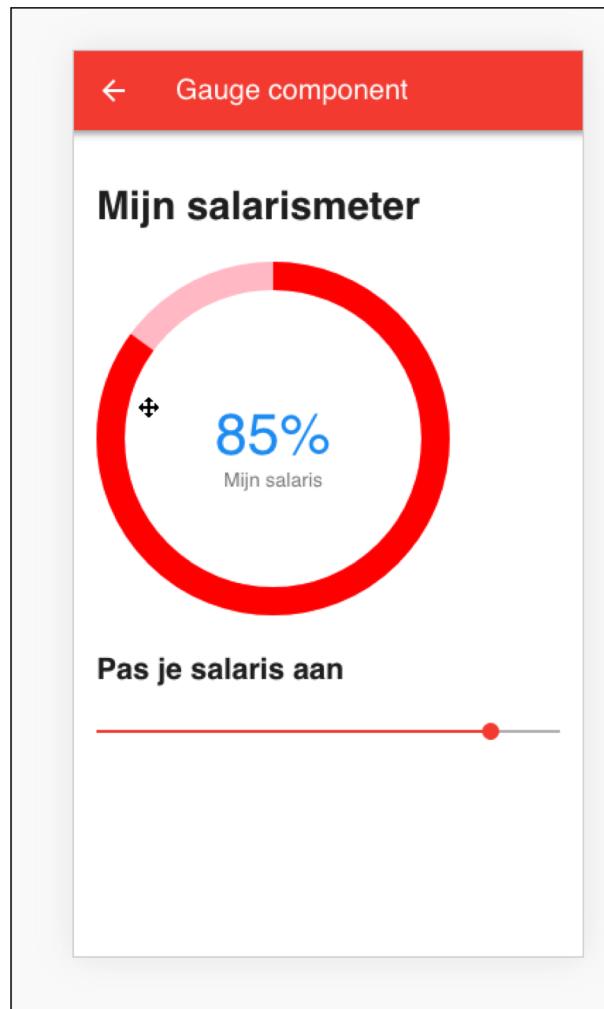
Code voor slider

- Introductie Dom7 – de jQuery-kloon
 - Dom7 wordt aangegeven met \$\$. Dat is een conventie
- Referentie naar object ophalen, callbackfunctie schrijven

```
// Selecteer de slider en voeg functionaleit voor 'change' toe
$$('#salaris-slider').on('range:change', function (e, range) {
    console.log('Er is een wijziging in de range', range.value);

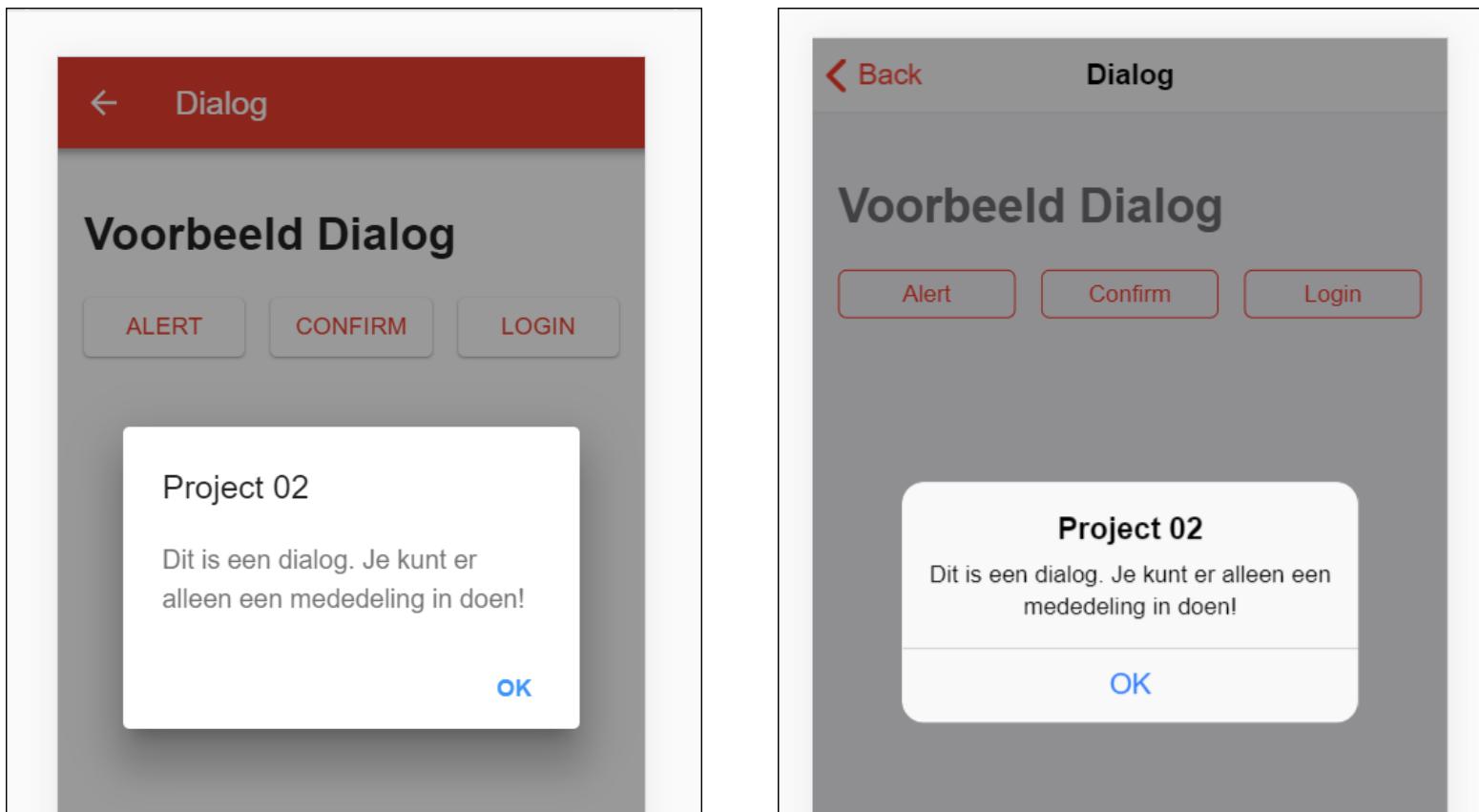
    // Ga de gauge updaten
    gauge.update({
        value: (range.value / 100),
        valueText: range.value + '%',
    });
})
```

Resultaat



Voorbeeld Dialog

- Dialog – een standaard OS-popup met melding: bericht, bevestiging, vraag, etc.
- Past zich automatisch aan aan Android/iOS



Binnen een template: @notatie voor aanroepen van functies:

```
<button class="col button button-raised" @click="openAlert">  
    Alert  
</button>
```

Onder de template: <script> voor uitbreiden intern JavaScript-object

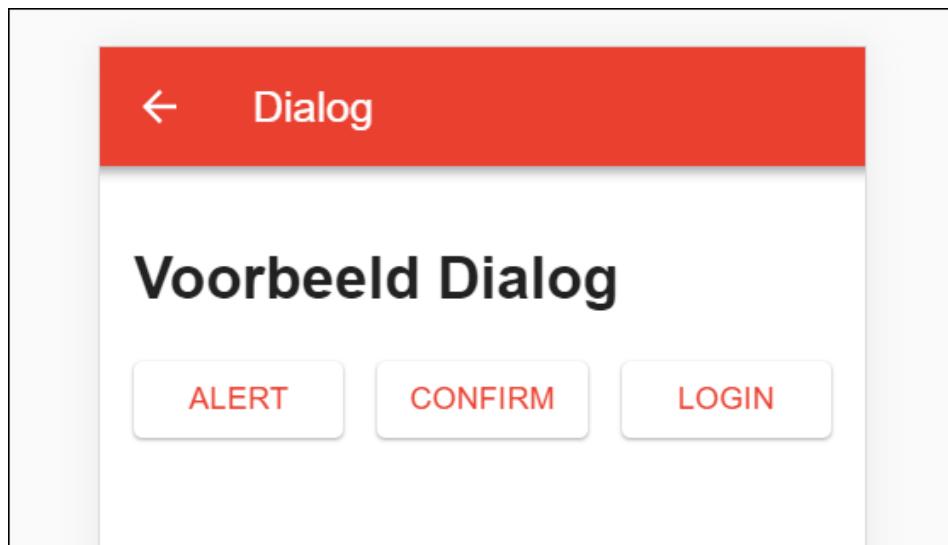
```
<script>  
    return {  
        methods: {  
            // standaard alert-venster  
            openAlert: function () {  
                app.dialog.alert('Dit is een dialog... ');  
            },  
            ...  
        }  
    }  
</script>
```



Standaard alert dialog

Dialog.html

- Drie knoppen, drie functies
- Bekijk de code van dialog.html



Beschikbare shortcuts:

- Alert
- Confirm
- Prompt
- Login
- Password
- Preloader
- Progress

Custom Dialog

- Zelf de inhoud van dialog samenstellen:
 - Stel parameters in voor de dialog (titel, inhoud, knoppen etc)
 - Maak een variabele

```
var customDialog = app.dialog.create(parameters)
```

- Open de dialog via een @click functie

```
openCustom: function () {  
    // zelfgemaakte dialog openen  
    customDialog.open();  
}
```

Dialog App Methods

Lets look at related App methods to work with Dialog:

`app.dialog.create(parameters)`- create Dialog instance

- `parameters` - `object`. Object with dialog parameters

Method returns created Dialog's instance

`app.dialog.destroy(el)`- destroy Dialog instance
• `el` - `HTMLElement` or `string`
`app.dialog.get(el)` - get Dialog instance by element

Dialog Parameters

Now lets look at list of available parameters we need to create Dialog:

| Parameter | Type | Default | Description |
|-----------------------------------|--------------------------|-------------------|---|
| <code>el</code> | <code>HTMLElement</code> | | Dialog element. Can be useful if you already have Dialog element create new instance using this element |
| <code>backdrop</code> | <code>boolean</code> | <code>true</code> | Enables Dialog backdrop (dark semi transparent layer behind) |
| <code>closeByBackdropClick</code> | <code>boolean</code> | | |
| <code>animate</code> | <code>boolean</code> | | |
| <code>title</code> | <code>string</code> | | |
| <code>text</code> | <code>string</code> | | |
| <code>content</code> | <code>string</code> | | |
| <code>buttons</code> | <code>array</code> | | |

Button Parameters

Each Button in `buttons` array must be presented as object with button parameters:

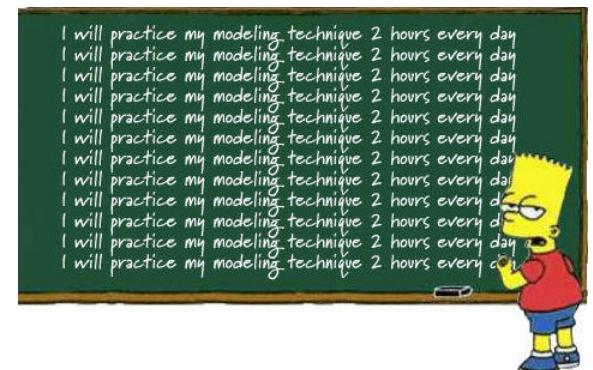
| Parameter | Type | Default | Description |
|-----------------------|----------------------------------|--------------------|---|
| <code>text</code> | <code>string</code> | | String with Button's text (could be HTML string) |
| <code>bold</code> | <code>boolean</code> | <code>false</code> | Enables bold button text |
| <code>color</code> | <code>string</code> | | Button color, one of <code>default colors</code> |
| <code>close</code> | <code>boolean</code> | <code>true</code> | If enabled then button click will close Dialog |
| <code>cssClass</code> | <code>string</code> | | Additional button CSS class |
| <code>keyCodes</code> | <code>array</code> | <code>[]</code> | Array with keyboard keycodes that will be used to trigger button click. It means that button click will be triggered on Enter key |
| <code>onClick</code> | <code>function(dialog, e)</code> | | Callback function that will be executed after click on button |

Dialog Methods & Properties

<https://framework7.io/docs/dialog.html>

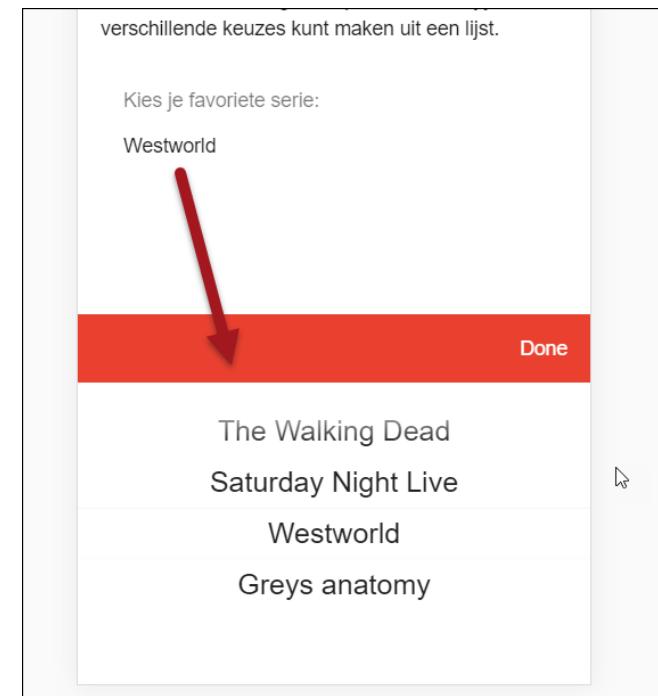
Oefening

- Bekijk het voorbeeld dialog.html
- Voeg zelf een knop en -code toe voor het maken van een prompt() dialoog
 - Toon de getypte tekst met console.log()
- Uncomment en bekijk het voorbeeld customDialog
 - Voeg zelf een knop toe en maak zelf een dialog die met deze knop wordt geopend.



Voorbeeld Picker

- Vaak gebruikt: kiezen uit een lijst
 - Namen,
 - Datums,
 - Lijst met meerdere opties
 - ...
- Component: Picker, met diverse data.



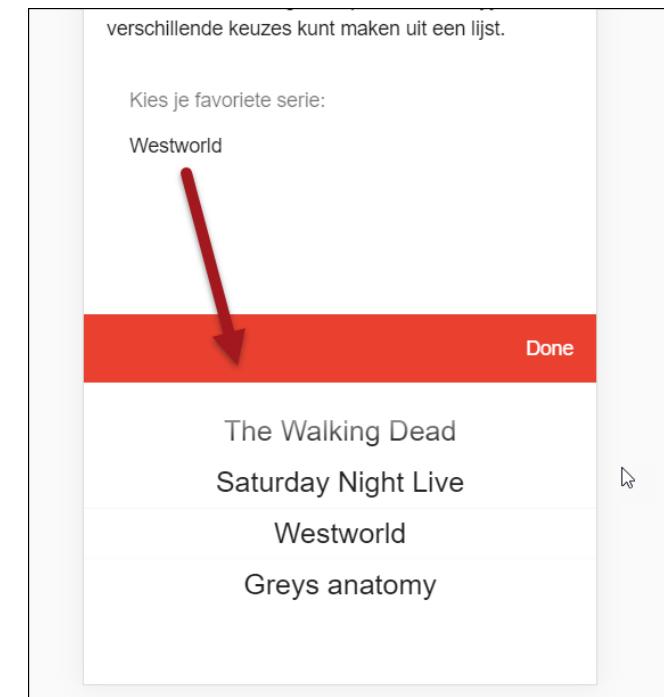
Werkwijze Picker

- Grotendeels vergelijkbaar - maak een template
- Maak een element dat als picker moet worden opgemaakt
 - Geef het element een `id`
- Schrijf script
 - Maak een object met extra methodes en data om de `picker` te creeren.

```
<input type="text" placeholder="serie"
       readonly="readonly" id="picker-serie" />
```

script

```
return {
  on: {
    pageInit: function (e, page) {
      var self = this; // caching van variabele 'this'
      // Series Device picker
      self.pickerSerie = app.picker.create({
        inputEl: '#picker-serie',
        cols: [
          {
            textAlign: 'center',
            values: ['House of Cards', ...]
          }
        ]
      });
    },
  }
}
```

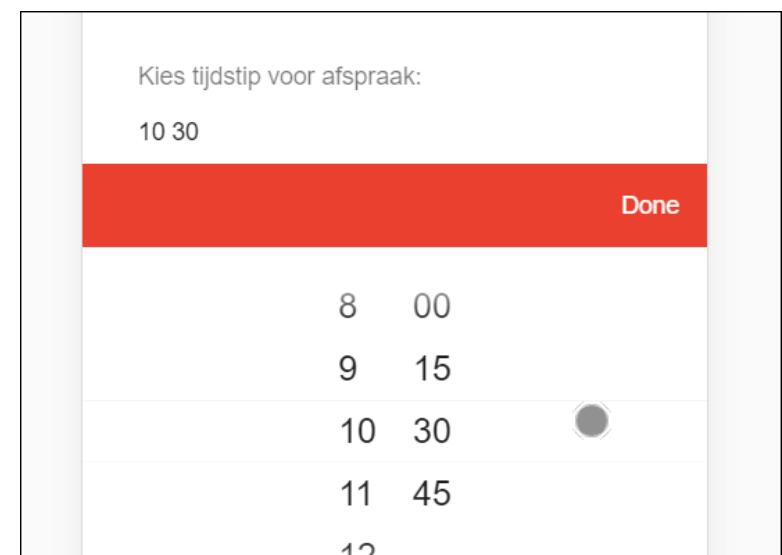


Picker met twee waarden

Geef extra kolom waarden op, bijvoorbeeld op deze wijze:

```
<input type="text" placeholder="tijd..." readonly="readonly"  
      id="picker-afspraak" />
```

```
self.pickerAfspraak = app.picker.create({  
  inputEl: '#picker-afspraak',  
  cols: [  
    {  
      textAlign: 'left',  
      values: ['8', '9', '10', ...]  
    },  
    {  
      values: ['00', '15', '30', '45']  
    }  
  ]  
})
```



Extra JavaScript – knop ‘Bevestig’

Zelf aanvullende HTML en JavaScript schrijven om iets met de ingevoerde waarden te doen

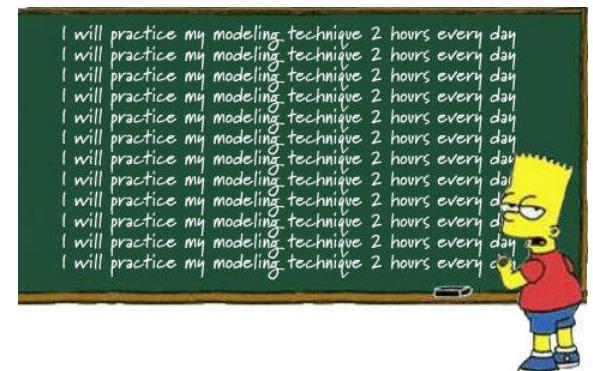
```
<button class="button button-raised" id="btnBevestig">  
    Bevestigen  
</button>
```

Dom7, de ‘Jquery’
van Framework7

```
// Knop Bevestig  
var $$ = Dom7; // referentie  
var btnBevestig = $$('#btnBevestig');  
var pickerSerie = $$('#picker-serie')  
var pickerAfspraak = $$('#picker-afspraak')  
  
btnBevestig.on('click', function () {  
    app.dialog.alert(  
        'Serie: ' + pickerSerie.val() + '<br>Afspraak: ' + pickerAfspraak.val(),  
        'Samenvatting')  
    .open();  
})
```

Oefening

- Bekijk het voorbeeld picker.html
- Voeg een picker toe waarbij de bezoeker zijn favoriete fruitsoort kan kiezen.
- Toon de fruitsoort in de dialog als de bezoeker op Bevestig klikt
- Bekijk mogelijk andere soorten pickers in de Kitchen Sink
- Meer info: <https://framework7.io/docs/picker.html>



Meer dynamische componenten

- Maak een popover
 - Popover lijkt op een dialog, maar wordt automatisch gesloten als buiten de popover wordt geklikt.
 - <https://framework7.io/docs/popover.html>
- Maak een toast
 - Toast is een korte melding die na enige tijd vanzelf weer verdwijnt.
 - Optioneel : knoppen om de toast zelf te sluiten
 - <https://framework7.io/docs/toast.html>
- Bekijk andere dynamische componenten naar wens

