

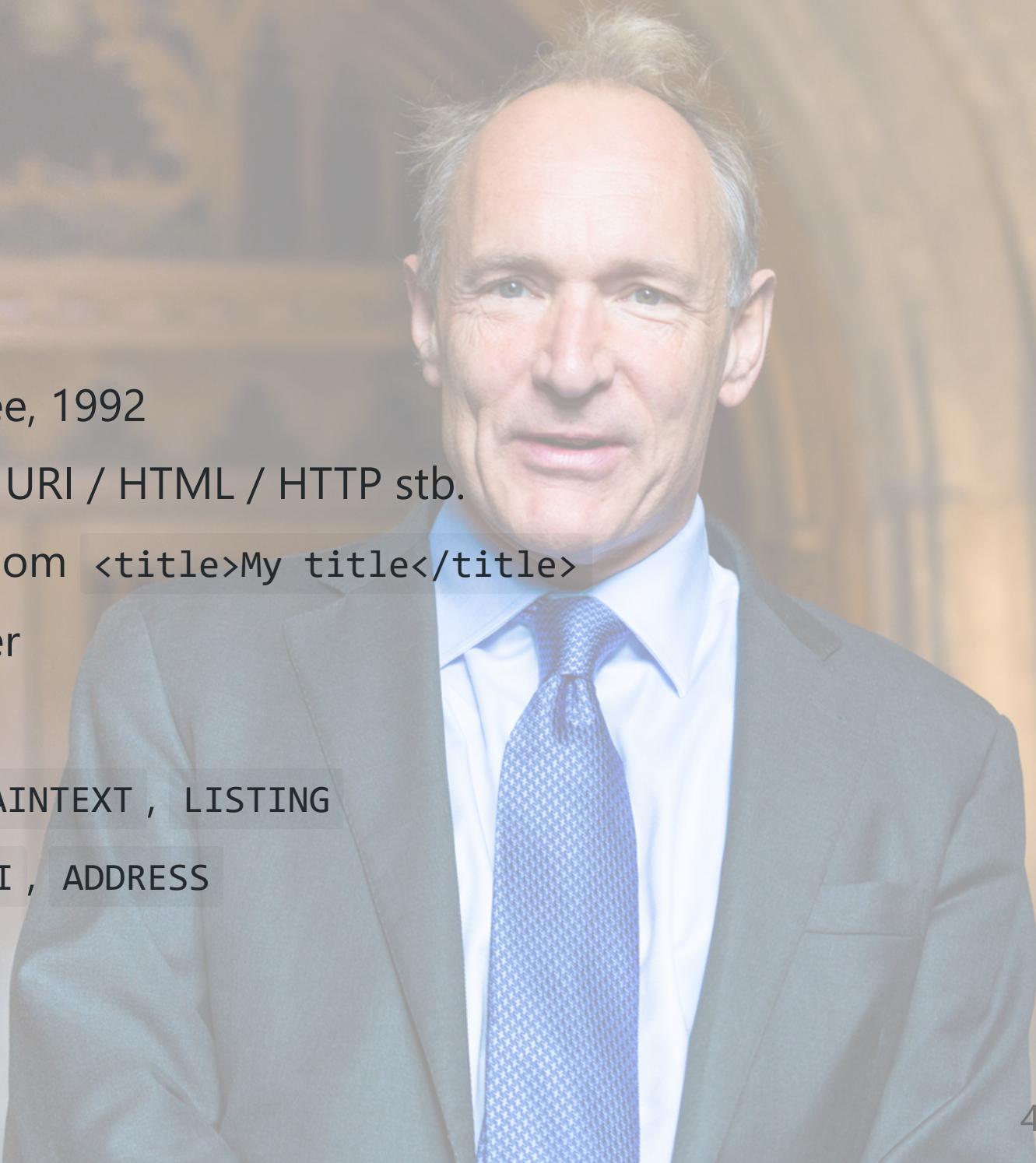
# Komponens alapú webfejlesztés

Polymer alapokon

Sándor Zsolt

Minden jog fenntartva.

# Rövid webtörténelem

A portrait photograph of Tim Berners-Lee, a man with light-colored hair, wearing a grey suit, white shirt, and blue patterned tie. He is looking slightly to his left with a neutral expression. The background is blurred, showing what appears to be a large hall or auditorium.

# HTML 1.0

- Tim Berners-Lee, 1992
- W3C alapítója, URI / HTML / HTTP stb.
- TAG mint fogalom <title>My title</title>
- Első webszerver
- Első böngésző
- TITLE , A , PLAINTEXT , LISTING
- P , H1 , UL , LI , ADDRESS

# HTML 2

- 1994
- FORM , INPUT , TEXTAREA , BR , CITE , CODE , STRONG
- HEAD , META , IMG

# HTML 3.2

- 1995
- Tartalom és megjelenés különválása
- DIV
- STYLE : CSS (Cascading style sheets)
- SCRIPT : JavaScript (Netscape Navigator 2.0)

# HTML 4

- 1997
- BUTTON , FRAME , OBJECT , SPAN
- CSS támogatás mindenhol

# HTML5

- 2014
- Semantics
  - `article` , `aside` , `footer` , `details` , `nav` , ...
- Komponensek
  - `audio` , `video` , `dialog` , `meter` , `date` , `range` , ...
- API:
  - `canvas` , `history` , `offline` , `storage` , ...

# Mi a komponens?

- Adott rendszer (vagy keretrendszer) részét képező
- Különálló
- Általános célú elem
- Szabadon beépíthető (újrafelhasználható)
- Önmagában zárt egységet képez

# Mi a webkomponens?

- Grafikus elem
- Web oldalon jelenik meg

# "Komponensek"

- HTML kód
- CSS
- Copy-paste
- Szerver oldal: sablonok

# Problémák

- Karbantarthatóság
- Újrafelhasználhatóság
- CSS ütközés

```
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

```
<bootstrap-card title="Card title" button-label="Go somewhere"  
    text="Some quick example text" ...>  
</bootstrap-card>
```

# Könyvtárak/megoldások

- VueJS
- React
- Riot
- Knockout
- Polymer

# Közös funkciók

- Komponens létrehozása
- Templating
- Styling
- Data binding/event handling
- Routing
- Komponens könyvtár
- Tesztelés

# Amit mi szerettünk volna

- Egyszerű
- Támogatott
- Javascript könyvtár
- Egyedi komponensek fejlesztéséhez
- Minél közelebb a böngésző funkcióihoz

# Miért Polymer?

- Google
- Böngésző funkciókra épül
- Rengeteg komponens ([webcomponents.org](http://webcomponents.org))
- Jó community (Slack)

# Mi a Polymer?

- Kis méretű (46->10k között, polyfilltől függően)
- Nyílt forráskódú (Google BSD)
- Javascript könyvtár
- Web Components szabványokra épül
- #UseThePlatform

# Hol használják?



# PREDIX



# Web Components

- HTML5 szabvány
- Web platform API funkciók
- Egyedi
- Újrafelhasználható
- Egységbe zárt HTML tagek létrehozásához

# Építőkövek

- HTML Templates
- Shadow DOM
- Custom Elements
- HTML Imports

# HTML Templates: cél

- JavaScript kódból
- Dinamikusan beilleszthető
- HTML sablon

# HTML Templates

- HTML tag: <template>
- Kliens oldali tartalmat tárol
- Oldalbetöltéskor csak feldolgozódik
- Nincs azonnali megjelenítés
- Futásidőben hozzuk létre JavaScript-ből

```
<template id="my-template">
  <h1>Hello world from template!</h1>
</template>

<div id="parent">
</div>

<script>
  let template = document.getElementById("my-template")
    .content.cloneNode(true);

  document.getElementById("parent").appendChild(template)
</script>
```

## HTML Imports: cél

- Meglévő sablonok kiszervezése külső fájlba
- Automatikus betöltés

# HTML Imports

- HTML dokumentumok beolvasása más HTML dokumentumból
- Csak elemzés (parsing)
- Nem csak HTML, hanem CSS és JavaScript is
- Javascript-ből lekérdezhető a betöltött tartalom
- Majd beszúrható a fő dokumentumba

# Betöljük a külső html kódot

```
<head>
  <link rel="import" href="child.html">
</head>
```

# Hozzáadjuk a dokumentumhoz

```
<div id="wrapper"></div>

<script>
  var link = document.querySelector('link[rel="import"]');
  var content = link.import;
  var el = content.querySelector('.part');
  var wrapper = document.querySelector("#wrapper");
  wrapper.appendChild(el.cloneNode(true));
</script>
```

## Shadow DOM: cél

- Egysége zárt HTML + CSS úgy, hogy
- Nincs hatással a fő dokumentumra
- Fő dokumentum se hat rá

# Shadow DOM

- Weboldal elemeit külön csomagoljuk és
- Leválasszuk a fő oldalstruktúrától
- CSS és egyéb kiegészítőkkel együtt

```
<div id="wrapper"></div>

<script>
    let shadowRoot = document.getElementById("wrapper")
        .attachShadow({
            mode: 'open'
        });
    shadowRoot.innerHTML = '<h1>And some header</h1>';
    shadowRoot.innerHTML += '<div>This text is red</div>';
    shadowRoot.innerHTML += '<style>div { color: red; }</style>
</script>
```

# Eredmény

```
<div id="wrapper">
  #shadow-root (open)
    <h1>This is some header</h1>
    <div>This text is red</div>
    <style>div {color: red; }</style>
</div>
<div>This has normal color.</div>
```

## **Custom Elements: cél**

- Saját, egyedi TAG a kódok helyett

# Custom Elements

- Egyedi TAG létrehozására:
  - Leszármaztatás ( `HTMLElement` )
  - Regisztráció
  - Használat

# Elem létrehozása

```
class HelloWorld extends HTMLElement {
    constructor() {
        super();

        let shadowRoot = this.attachShadow({
            mode: 'open'
        });

        let label = document.createElement("span");
        label.textContent = "Hello world";
        shadowRoot.appendChild(label);
    }
}
```

# Regisztráció

```
customElements.define("hello-world", HelloWorld);
```

# Custom element használata

```
<hello-world></hello-world>
```

## Custom element dinamikus használata

```
let helloworld = new HelloWorld();
let placeholder = document.querySelector('#placeholder');

placeholder.appendChild(helloworld);
```

# Polyfill

- Újabb web szabványok implementálása
- Korábbi standardok és
- JavaScript segítségével
- Amit a régebbi böngészők nem támogatnak

# Polymer

- WebComponents +
  - Property - Attribute binding
  - Data binding
  - CSS mixin
  - CLI
  - Tesztkörnyezet
  - Könyvtár

# Az első komponensünk

# A cél

2016 06/06  
Mon

May		June					July	
S	M	T	W	T	F	S		
29	30	31	1	2	3	4		
5	6	7	8	9	10	11		
12	13	14	15	16	17	18		
19	20	21	22	23	24	25		
26	27	28	29	30	1	2		

Wed June 6th , 2016 •  
10:55 AM

Live in the present moment

Without question, many of us have mastered the neurotic art of spending much of our lives worrying about a variety of things — all at once We allow past problems and future concerns to dominate our present moments, so much so that we end up anxious, frustrated, depressed, and hopeless.

+

2016 06/06  
Mon

May		June					July	
S	M	T	W	T	F	S		
5	6	7	8	9	10	11		



Wed June 6th , 2016 •  
10:55 AM

Live in the present moment

Without question, many of us have mastered the neurotic art of spending much of our lives worrying about a variety of things — all at once We allow past problems and future concerns to dominate our present moments, so much so that we end up anxious, frustrated, depressed, and hopeless.

+

Minden jog fenntartva.

# Megvalósítandó funkciók

- Aktuális nap mutatás
- Naptár
  - Előző hónap napjai a héten
  - Adott hónap napjai
  - Előző hónap
  - Következő hónap
  - Nap választás

# Függőségek telepítése

- <https://nodejs.org/en>
- `npm install -g polymer-cli`
- `npm install -g bower`

# Könyvtárstruktúra létrehozása

- `md [könyvtár]`
- `cd [könyvtár]`
- `polymer init`
  - `polymer-2-element` - A simple Polymer 2.0 template
  - Element name: `acme-calendar`
  - Brief description: An awesome calendar component...
- `polymer serve --open`

# acme-calendar.html

```
<link rel="import" href="../polymer/polymer-element.html">

<dom-module id="acme-calendar">
  <template>
    <style>
      :host {
        display: block;
      }
    </style>
    ...
  </template>

  <script>
    class AcmeCalendar extends Polymer.Element {
      static get is() { return 'acme-calendar'; }
      ...
    }
  }

  window.customElements.define(AcmeCalendar.is, AcmeCalendar)
</script>
</dom-module>
```

# Felépítés

- Import szekció
- Dom-module
  - Template
  - Script

## Step 2 - bemeneti paraméterek

- Bemeneti paraméterek
  - Definiálása
  - Megjelenítése

# Bemeneti paraméterek definiálása

- Év (year)
- Hónap (month)
- Nap (day)
- Attribútumon keresztül

# Mi az attribútum?

- Egy HTML tag tulajdonsága
- Forrásként szolgál
- name="value" formátum

## Példa

```
<acme-calendar year="2018" month="5" day="12">
```

# Mi a property?

- Polymer elem (class) tulajdonsága
- Rögzített típus
- Összekötődik az azonos nevű attribútummal
  - Összetett nevű property esetén kötőjel az attribútumban
  - `headerTitle` -> `header-title`

## Példa

```
class AcmeCalendar extends Polymer.Element {  
  
    static get is() {  
        return 'acme-calendar';  
    }  
  
    static get properties() {  
        return {  
            year: Number,  
            month: Number,  
            day: Number  
        };  
    }  
}
```

# Bemeneti paraméterek megjelenítése

- `template`-en belül
- `[[[]]]` vagy `{{{}}}` segítségével

```
<h2>Year: [[year]] Month: [[month]] Day: [[day]]</h2>
```

## Step 3

- Napok megjelenítése
  - Metódus hozzáadása
    - Hét napjainak listája
    - Hónap napjainak listája
  - Dinamikus tartalom generálása

# Metódusok hozzáadása

- Hagyományos JavaScript metódusok
- Aláhúzással kezdődik (`_myMethod`)

```
_getDays() {  
    return ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"];  
}  
  
_getDaysOfMonth(year, month) {  
    return Array.from({  
        length: 30  
    }, (x, i) => i + 1);  
}
```

# Dinamikus tartalom generálása

- Segéd tag-ek segítségével
- `dom-repeat` : lista megjelenítése
- `dom-if` : feltételes megjelenítés

# dom-repeat

- `<link rel="import" href="../polymer/lib/elements/dom-repeat.html">`
- Lista iterátor
- Menjünk végig egy lista minden elemén, és...
- `<template is="dom-repeat" items="{{...}}>`
- Ezen belül `item`-ként lehet hivatkozni az egyes elemekre
- Alternatív elnevezés: `as="something"`

# Példa

```
<template is="dom-repeat" items="[_getDays()]" as="day">  
  [[day]]  
</template>
```

```
class AcmeCalendar extends Polymer.Element {  
  ...  
  _getDays() {  
    return ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]  
  }  
  ...  
}
```

## dom-if

- `<link rel="import" href="../polymer/lib/elements/dom-if.html">`
- Feltétel vizsgálat
- Kifejezés esetén metódushívás
- Else nincs :(

```
<template is="dom-if" if="[[visible]]">  
  This is now displayed...  
</template>
```

```
<template is="dom-if" if="[_isVisible(age, 23)]">  
    This is displayed only if age is less than 23 displayed...  
</template>
```

## Step 4

- Div struktúrák hozzáadása

# Step 5

- CSS hozzáadása
- Dinamikus attribútumok

# Dinamikus attribútumok

- `attr = '[[...]]'` vagy `attr = '{{..}}'`
- `element.attr = value;`
- Kivétel
  - `class`, `style`, `href`, `for`, `data-*`
  - `value` (`<input type="number">`)
  - Jelölés: `$=`
  - `element.setAttribute(attr,value)`

```
<my-item name="[_shorten(name)]"></my-item>
<div class$="day [_dayClass(item, day)]">
    [[item]]
</div>
```

## Step 6

- Külső JavaScript könyvtár használata
- Kód szeparáció: mixin
- CSS styling

# **Step 6 - külső JavaScript könyvtár**

# Moment

- <https://momentjs.com>
- bower install moment --save

## shared-moment.html

- HTML Import
- Példa:

```
<script src="bower_components/moment/min/moment-with-locales.mi
<script src="bower_components/moment-timezone/moment-timezone.j
<script src="bower_components/moment-timezone/builds/moment-tim
```

## **Step 6 - Kód szeparáció**

# Mixin

- Függvénykönyvtár
- Kód duplikálás elkerülésére
- `<link rel="import" href="mixin-datetime.html">`

```
<script>
  window.acme = window.acme || {};
  acme.DatetimeMixin = Polymer.dedupingMixin((base) => class
    _myMixinMethod(){
      console.log("Here comes my method");
    }
  );
</script>
```

# Mixin használat

```
class AcmeCalendar extends Acme.DatetimeMixin(Polymer.Element){  
  ...  
  _myHostMethod(){  
    ...  
    this._myMixinMethod();  
    ...  
  }  
}
```

## **Step 6 - styling**

```
:host {  
    display: block;  
    --row-height: 60px;  
    --current-day-background: #10ACFF;  
    --current-day-color: white;  
}
```

```
#myCalendar{  
    --current-day-background: red;  
    --current-day-color: blue;  
}  
  
<acme-calendar id="myCalendar"></acme-calendar>
```

# CSS változók

- Mint egy normál változó
- -- el kell kezdődnie
- definíció a `:host`-ban
- használat: `var(--my-variable)`

```
:host{  
  --current-day-background: #10ACFF;  
  --current-day-color: white;  
}  
  
.currentDay {  
  background-color: var(--current-day-background);  
  color: var(--current-day-color);  
  border-radius: 30px;  
}
```

# CSS változó alapérték

```
.my-image {  
    border-radius: var(--my-custom-radius, 3px);  
}
```

# CSS változó modosítása kódból

```
this.updateStyles({  
  '--my-custom-radius': '30px',  
});
```

# CSS mixin

```
:host{  
  --my-mixin: {  
    background: #256dbd;  
    color: #f5f5f5;  
  }  
}  
  
#myDiv{  
  @apply --my-mixin;  
}
```

## Step 7 - Eseménykezelés

- Eseményre figyelés
- on-event attribútum (pl. on-click )
- Mindig kisbetű

```
<div on-click="_onDivClicked">Click me..</div>  
...  
_onDivClicked(event) {  
    // ...  
}
```

## Dinamikus eseményre figyelés

- Lifecycle metódusokban
- `this.addEventListener('event', function)`
- `this.addEventListener('click', this._onClick)`

```
ready() {
    super.ready();
    this.addEventListener('click', this._onClick);
}

_onClick(event) {
    // ...
}
```

## Step 8 - állapot kimenet

- Esemény
- Kétirányú data binding

# Esemény küldése

- `dispatchEvent(new CustomEvent(...))`
- `bubbles` : külső komponensnek továbbdobja-e
- `composed` : shadow-dom határokon át
- `detail` : adat

```
let event = new CustomEvent("calendar-changed", { bubbles:  
    composed: true,  
    detail : { year : year, month : month, day: day}  
});  
  
this.dispatchEvent(event);
```

# Esemény fogadása

```
<acme-calendar on-calendar-changed="_handleEvent"></acme-ca  
...  
  
_handleEvent(event){  
    const detail = event.detail;  
    this.year = event.detail.year;  
}
```

# Observer definíció

- Több property változás együtt

```
static get observers() {  
    return [  
        '_updateDate(year, month, day)'  
    ]  
}
```

```
_updateDate(year, month, day) {
  if (year && month && day) {
    const event = new CustomEvent("calendar-changed", {
      bubbles: true,
      composed: true,
      detail: {
        year: year,
        month: month,
        day: day
      }
    });
    this.dispatchEvent(event);
  }
}
```

# Kétirányú data binding

- `notify: true`
- Egy eseményt küld ha a property értéke megváltozik
- Esemény neve: `property-name-changed`

# Kétirányú data binding

```
year: {  
    type: Number,  
    notify: true  
}
```

# Kétirányú data binding

```
<h1>Year: [[year]]</h1>
<h1>Month: [[month]]</h1>
<h1>Day: [[day]]</h1>

<acme-calendar year="{{year}}" month="{{month}}" day="{{day}}"
```

## **Step 9 - tesztelés**

# Tesztelő modulok

- Mocha: test framework
- Chai: assertion
- Sinon: stub, spy, mock
- Selenium: tesztelés különböző böngészőkkel

# Tesztkörnyezet telepítés

- `bower install --save-dev web-component-tester`

# Tesztkörnyezet futtatása

- polymer test
- polymer test -l chrome
- polymer serve
- localhost:8080/components/acme-calendar/test/acme-calendar\_test.html

# Test fixture

```
<test-fixture id="ChangedPropertyTestFixture">
  <template>
    <acme-calendar year="2015" month="5" day="13"></acm
  </template>
</test-fixture>
```

# Test JavaScript code

```
suite('acme-calendar', () => {  
  test('here comes the test name', () => {  
    ...  
  });  
});
```

# Test JavaScript code

```
suite('acme-calendar', () => {  
  test('here comes the test name', () => {  
    const element = fixture('ChangedPropertyTestFix');  
    const elementShadowRoot = element.shadowRoot;  
  
    assert(...);  
  });  
});
```

# Breakpoint

- `debugger;` utasítás

```
suite('acme-calendar', () => {  
  test('here comes the test name', () => {  
    debugger;  
  
    const element = fixture('ChangedPropertyTestFix');  
    const elementShadowRoot = element.shadowRoot;  
  
  });  
});
```

# Szinkron teszt

```
suite('acme-calendar', () => {

    test('here comes the test name', () => {

        debugger;

        const element = fixture('ChangedPropertyTestFix');
        const elementShadowRoot = element.shadowRoot;

        const template = elementShadowRoot.querySelector('#template');
        template.render();

        const selectedDay = elementShadowRoot.querySelector('.selected');
        assert.isDefined(selectedDay);

        const selectedDayText = selectedDay.textContent;
        assert.equal(selectedDayText, '13');

    });
});
```

# Aszinkron teszt

```
element.addEventListener('dom-change', function(event) {  
    const shadowRoot = element.shadowRoot;  
    const selector = "#daysPanel .day.currentDay";  
  
    const selectedDay = shadowRoot.querySelector(selector);  
  
    if (selectedDay) {  
        assert.equal(selectedDay.textContent.trim(), '13');  
        done();  
    }  
});  
  
element.setProperties({  
    year: '2015',  
    month: '5',  
    day: '13'  
});
```

# A jövő

- Polymer 3
- NPM support (yarn)
- ES6 modules
  - Chrome támogatja
  - A többieknél polyfill
  - Firefox about:config -> dom.moduleScripts.enabled