# Web Components

Tomasz
Ducin

Tomasz Ducin
13-15 February 2019
Wrocław

# agenda

- WebComponents
  - Custom Elements
  - Design
    - Attributes
    - Properties
    - Events
  - Shadow DOM
- architecture
  - Micro frontends
  - Strangler Apps
  - Angular Elements
  - React WC
- exercises

//jsfiddle.net/tomasz_ducin/g3roay3a/embedded/result/

## Benefits

- style encapsulation
- fewer dependencies & prerequisites
- framework/library agnostic
- longer life components
  (no *rewriting frontend every 6 weeks*)

# The Standard

## The Standard

- Custom Elements

# The Standard

- Custom Elements
- Shadow DOM

## The Standard

- Custom Elements
- Shadow DOM
- ~~HTML Templates~~

# The Standard

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

# Implementations

## Implementations

# Implementations

## Implementations

SkateJS

## Implementations

JS

SkateJS

x-tag

## Custom Elements API

# Custom Elements API

- constructor()

# Custom Elements API

- constructor()
- connectedCallback()

# Custom Elements API

- constructor()
- connectedCallback()
- disconnectedCallback()

# Custom Elements API

- constructor()
- connectedCallback()
- disconnectedCallback()
- attributeChangedCallback(
  attributeName, oldValue, newValue)

# Custom Elements API

- constructor()
- connectedCallback()
- disconnectedCallback()
- attributeChangedCallback(
  attributeName, oldValue, newValue)
- adoptedCallback(
  oldDocument, newDocument)

# data flow

props & callbacks

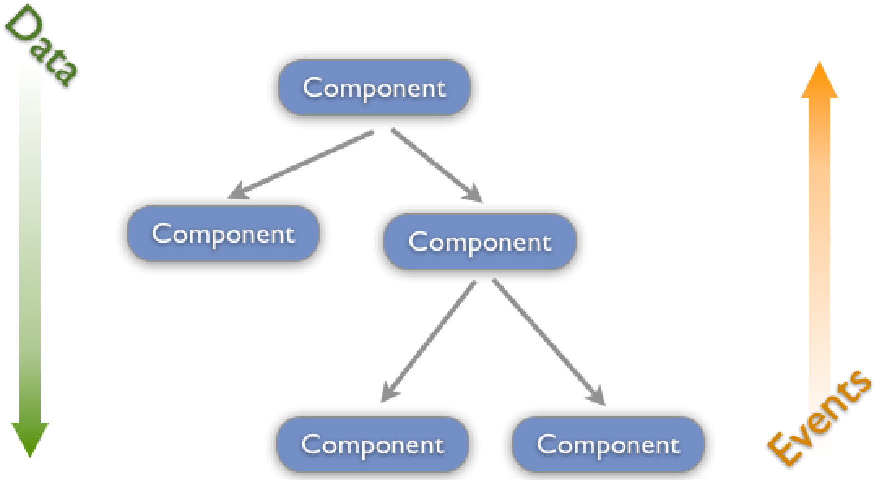
event streams, @Input, @Output

**attributes**

(strings only)

**properties**

(objects: arrays, functions, etc)



**native events**

**custom events**

no built-in render method

## Attributes

- `this.getAttribute('account')`
- `this.setAttribute('account', number)`
- `this.innerHTML`

## Attributes Updates

```
class AccountElement extends HTMLElement {

  static get observedAttributes() {
    return ['account'];
  }

  attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == 'account') {
      this.textContent = `Account no. ${newValue}`;
    }
  }
}

customElements.define('account-element', AccountElement);
```

without observedAttributes defined, only initial value is consumed (but changes are ignored)

# Events

```
this.addEventListener('click', (event) => {
  if (changed(value) && someOtherConditions) {
    this.dispatchEvent(new Event("change"));
  }
})
```

```
this.addEventListener('click', (event) => {
  if (changed(value) && someOtherConditions) {
    this.dispatchEvent(new CustomEvent("cat", {
      detail: {
        hazcheeseburger: true
      }
    }));
  }
})
```

# passing data down

| attributes | properties |
|---|---|
| HTML-based | JS-based |
| strings | **both primitives** (strings, numbers, boolean, ...) **and objects** (arrays, functions, ...) |

# passing data down

| attributes | properties |
|---|---|
| HTML-based | JS-based |
| strings | **both primitives** (strings, numbers, boolean, ...) **and objects** (arrays, functions, ...) |

```
<tr-history header="Transactions History" />
```

```
// reference to custom node
$history = document.querySelector('tr-history')

// fetching data async & assigning it manually
API.getTransactions()
.then(items => $history.transactions = items);
```

# passing data down

| **attributes** | **properties** |
|---|---|
| HTML-based | JS-based |
| strings | **both primitives** (strings, numbers, boolean, ...) **and objects** (arrays, functions, ...) |

```
<tr-history header="Transactions History" />
```

```
// reference to custom node
$history = document.querySelector('tr-history')

// fetching data async & assigning it manually
API.getTransactions()
.then(items => $history.transactions = items);
```

**imperative!**

# passing data down

| attributes | properties |
|---|---|
| HTML-based | JS-based |
| strings | **both primitives** (strings, numbers, boolean, ...) **and objects** (arrays, functions, ...) |

```
<tr-history header="Transactions History" />
```

```
// reference to custom node
$history = document.querySelector('tr-history')

// fetching data async & assigning it manually
API.getTransactions()
.then(items => $history.transactions = items);
```

**imperative!**

serializing objects into JSONs and passing as attributes is wrong!

# Properties Updates

- in order to pass strings,
  use attributes
- in order to pass anything else,
  use properties
- in order to use properties, treat the Custom Element as a native JavaScript object
- and handle it imperatively

# no JSX!

## JSX

**declarative**

```
function ActionLink(props) {
  function handleClick(e) {
    e.preventDefault();
    console.log('clicked.');
  }

  return (
    <a href="#" onClick={handleClick}>
      Go to {props.url}
    </a>
  );
}
```

## Web Components

**imperative**

```
// outside custom element
const $history = document.querySelector("tr-history");
$history.transactions = [
  { id: 3162, type: 'deposit', amount: 10000, ...},
  { id: 4473, type: 'withdraw', amount: 47.39, ...},
  { id: 4782, type: 'deposit', amount: 10000, ...}
];

// inside custom element
this.innerHTML = `<div class="history"></div>`;
this.querySelector('.history').innerHTML =
  this._transactions.map(t => `<div>
    <span>${t.id}, ${t.type}, ${t.amount}</span>
  </div>`);

set transactions(items){
  if (items === this._transactions) return;
  this._transactions = items;
}
```

**properties are set via vanilla JS, ~~not HTML~~**

# Properties Updates

- property setters are primitive:
  - don't support *onPush* change (need to check if the object passed is the same reference as current one, manually)
  - don't support re-render (need to re-render manually)

# Shadow DOM

```
// Create shadow DOM
var shadow = document.querySelector('#container')
  .attachShadow({mode: 'open'});

// Add some text to shadow DOM
shadow.innerHTML = '<p>list of transactions below</p>';

// Add some CSS to make the text red
shadow.innerHTML += '<style>p { color: blue; }</style>'
```

## modes:

- **open** - able to manipulate
- **closed** - never changes after created (unlikely to be used)

# Shadow DOM & CSS

## Shadow DOM styling

## Custom Checkbox

//jsfiddle.net/tomasz_ducin/g3roay3a/embedded/result/

//jsfiddle.net/tomasz_ducin/sfo5gkw0/2/embedded/result/

https://goo.gl/mBv6mG