[Angular]

WEB-39140 Created by Elena Pogorelova 3 years ago
Updated by Piotr Tomiak a year ago

Visible to issue readers

`$event.target.value` not resolved in Angular template

1

Project	WebStorm
Type	Bug
State	Submitted
Assignee	Piotr Tomiak
Subsystem	TypeScript
Affected	No Affected version
Tester	No tester
Verified	No

WS-192.4205.48

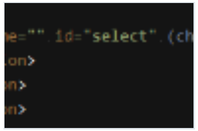
```
@Component({
  selector: 'app-keyup',
  templateUrl: './keyup.components.html',
  styleUrls: ['./keyup.component.css']
})
export class KeyUpComponent {
  onChange(value: string) {
    console.log('value: ' + value);
  }
}
```

template:

```
<label for="select"></label><select name="" id="select" (change)="onChange($event.ta
    <option value="">Select combo</option>
    <option value="Value1">Text1</option>
    <option value="Value2">Text2</option>
    <option value="Value3">Text3</option>
</select>
```

in `onChange($event.target.value)`, `value` is not resolved

Attachments 1



```
e="" id="select" (ch
on>
n>
n>
```

There is 1 attachment that is not visible to you

Piotr Tomiak commented 10 Aug 2020 19:17

The issue is due to typings specified in TS `lib.dom.d.ts` , which do not have dedicated event type for `onchange` in `HTMLSelectElement` interface, which is correct according to https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/change_event . The `change` event is mapped to generic interface `Event` , which has:

```
interface Event {
(...)
    readonly target: EventTarget | null;
(...)
}
```

and `EventTarget` (https://developer.mozilla.org/en-US/docs/Web/API/EventTarget) is specified as:

```
interface EventTarget {
    addEventListener(...): void;
    dispatchEvent(...): boolean;
    removeEventListener(...): void;
}
```

The event target can be any HTML Element and a cast to applicable interface should be applied here (like `HTMLSelectElement`), however Angular expressions doesn't allow casting. The best workaround would be to wrap `$event.target` with `$any()` :

```
<select (change)="onChange($any($event.target).value)">
```

Other workaround is to create a following shim to fix the `target` type:

shim.d.ts

```
declare interface HTMLSelectElement {
    onchange: (( this: GlobalEventHandlers, ev: SelectChangeEvent ) =>
}

interface SelectChangeEvent extends Event {
    readonly target: HTMLSelectElement | null;
}
```

In the latest 2020.2.1 WebStorm shim will not work, the support for shim will be present in 2020.3.

I will keep this issue open in case there is a better solution, which doesn't require any action from the user.