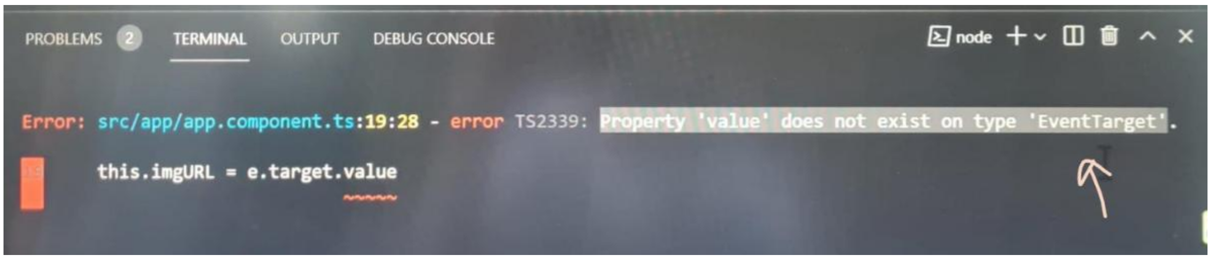
Type Assertions

https://picsum.photos/images

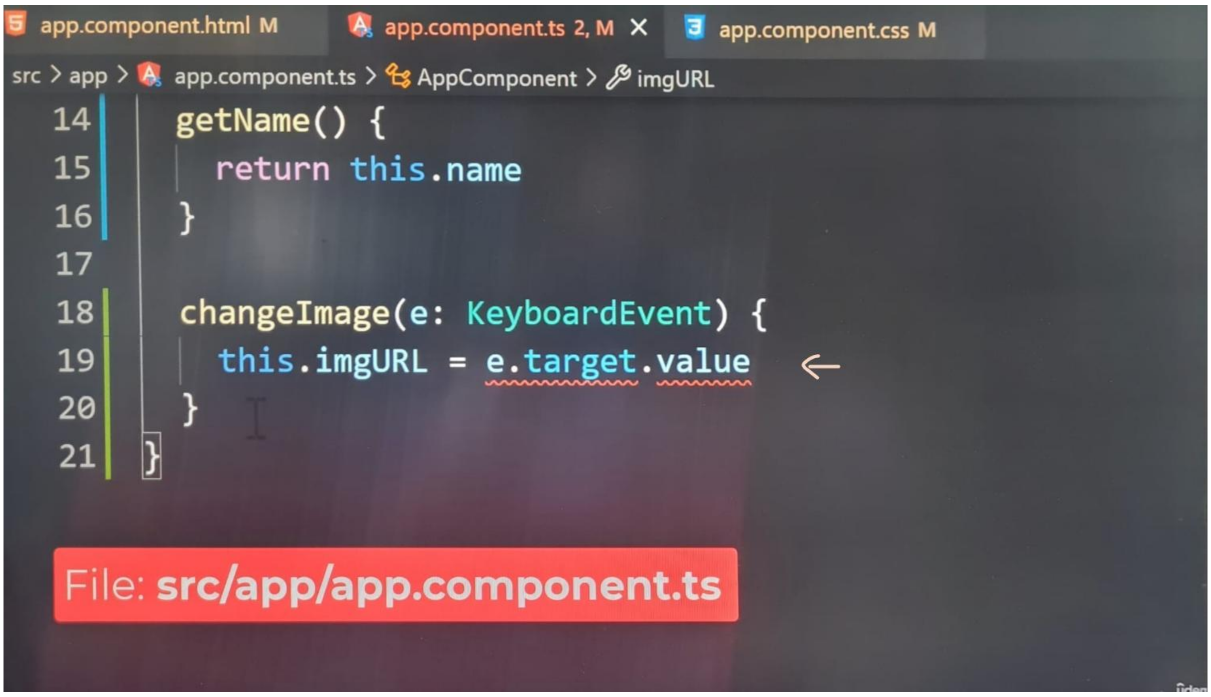
In this lecture, we're going to explore the error thrown by TypeScript continuing where we left off.



```
Error: src/app/app.component.ts:19:28 - error TS2339: Property 'value' does not exist on type 'EventTarget'.

      this.imgURL = e.target.value
```

Property: "value" does not exist on type "EventTarget".

Let's examine the issue, in our component, we are trying to update the image URL property to the **events.target.value** property:

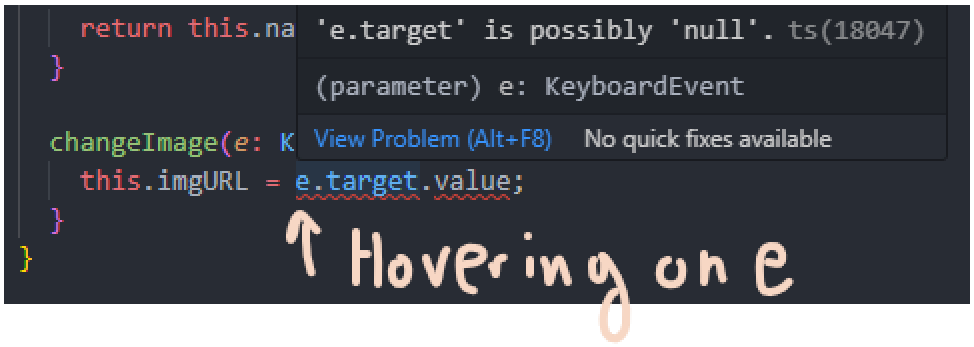

app.component.html M
app.component.ts 2, M
app.component.css M
src > app > app.component.ts > AppComponent > imgURL
File: src/app/app.component.ts

```
  getName() {
    return this.name
  }

  changeImage(e: KeyboardEvent) {
    this.imgURL = e.target.value
  }
}
```

According to the error, we're trying to access a non-existent property called value.

in regular JavaScript, the browser exposes the value of an input through the value property. Technically, if TypeScript weren't so strict, our code would work perfectly. So, what's the deal?

The issue we're encountering stems from TypeScript, not Angular. To better understand, let's hover our mouse over the e object:

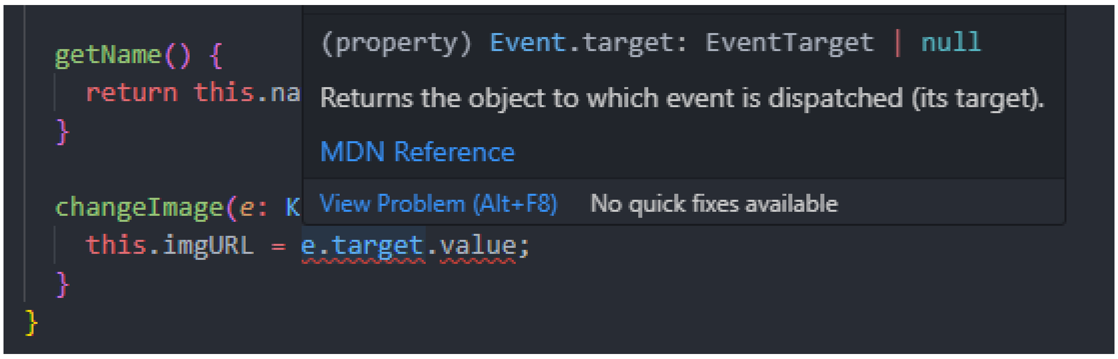


```
    return this.na
  }

  changeImage(e: K
    this.imgURL = e.target.value;
  }
}
```

```
'e.target' is possibly 'null'. ts(18047)
(parameter) e: KeyboardEvent
View Problem (Alt+F8)    No quick fixes available
```

Our editor will tell us the e object is annotated with the **KeyboardEvent** type.

Let's check out the data type of the target property:

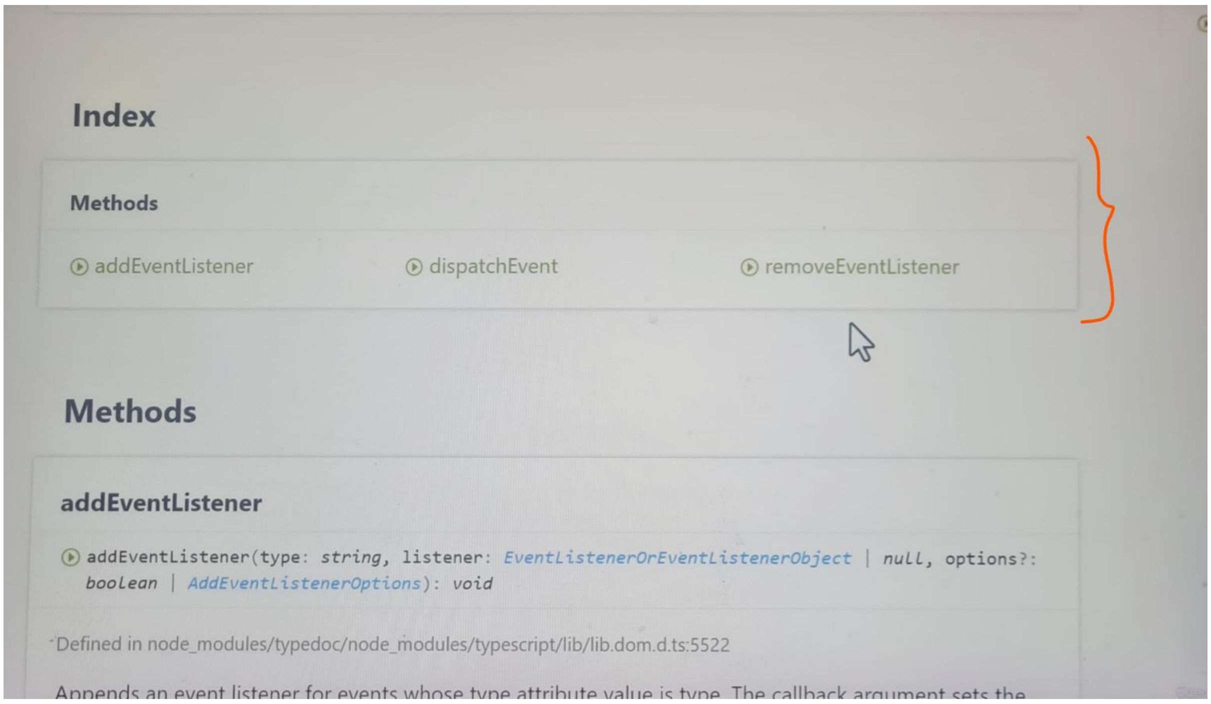


```
  getName() {
    return this.na
  }

  changeImage(e: K
    this.imgURL = e.target.value;
  }
}
```

```
(property) Event.target: EventTarget | null
Returns the object to which event is dispatched (its target).
MDN Reference
View Problem (Alt+F8)    No quick fixes available
```

This time, the editor is indicating the data type is set to **EventTarget**.

The **EventTarget Interface** (Provide link gives 404):



Index

Methods

addEventListener   dispatchEvent   removeEventListener

Methods

addEventListener

```
addEventListener(type: string, listener: EventListenerOrEventListenerObject | null, options?:
  boolean | AddEventListenerOptions): void
```

Defined in node_modules/typedoc/node_modules/typescript/lib/lib.dom.d.ts:5522

Appends an event listener for events whose type attribute value is type. The callback argument sets the

The index section contains a list of properties and methods, immediately, you'll realize the interface does not define a property called **value**. Hopefully, the error is starting to become apparent. TypeScript isn't aware there's a property called **value** on **target object**. This behaviour can be helpful, because it can prevent us from accessing properties that don't exist on an object. However, the **value** property does exist on the **target object**. Input elements expose the field's **value** through the **value property**. So why did the development team decide not to add the **target property** to the interface?

Keyboard Events can be emitted on various elements, they're not exclusive to input elements, if a **keyboard event** was emitted on a non-input element, the **value property** would be **undefined**. For this reason, the interface does not guarantee the **value property** will always exist on the **target object.**

This type of problem can appear from time to time, sometimes a property or variable can be annotated with an incomplete interface. Luckily, TypeScript offers a feature called **TypeAssertion**.

TypeAssertion allows us to change the type from the compilers perspective.

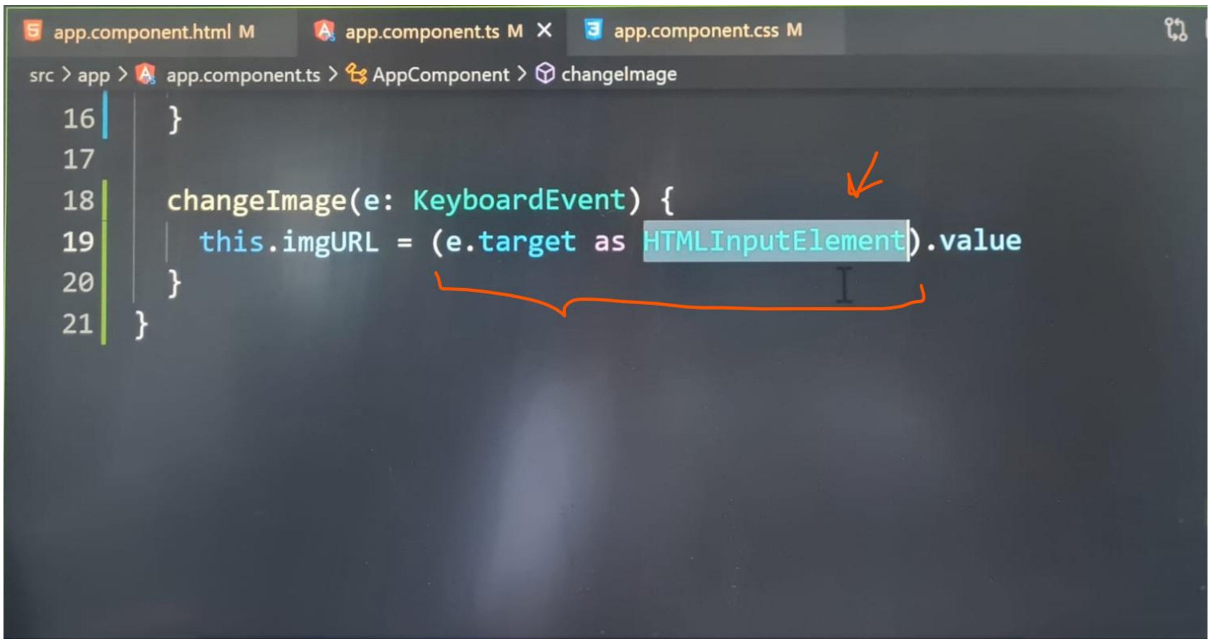
We can fix this error by wrapping the **e.target** property with parentheses () → **(e.target).**

Next, we need to add the following:

(e.target as HTMLInputElement).value

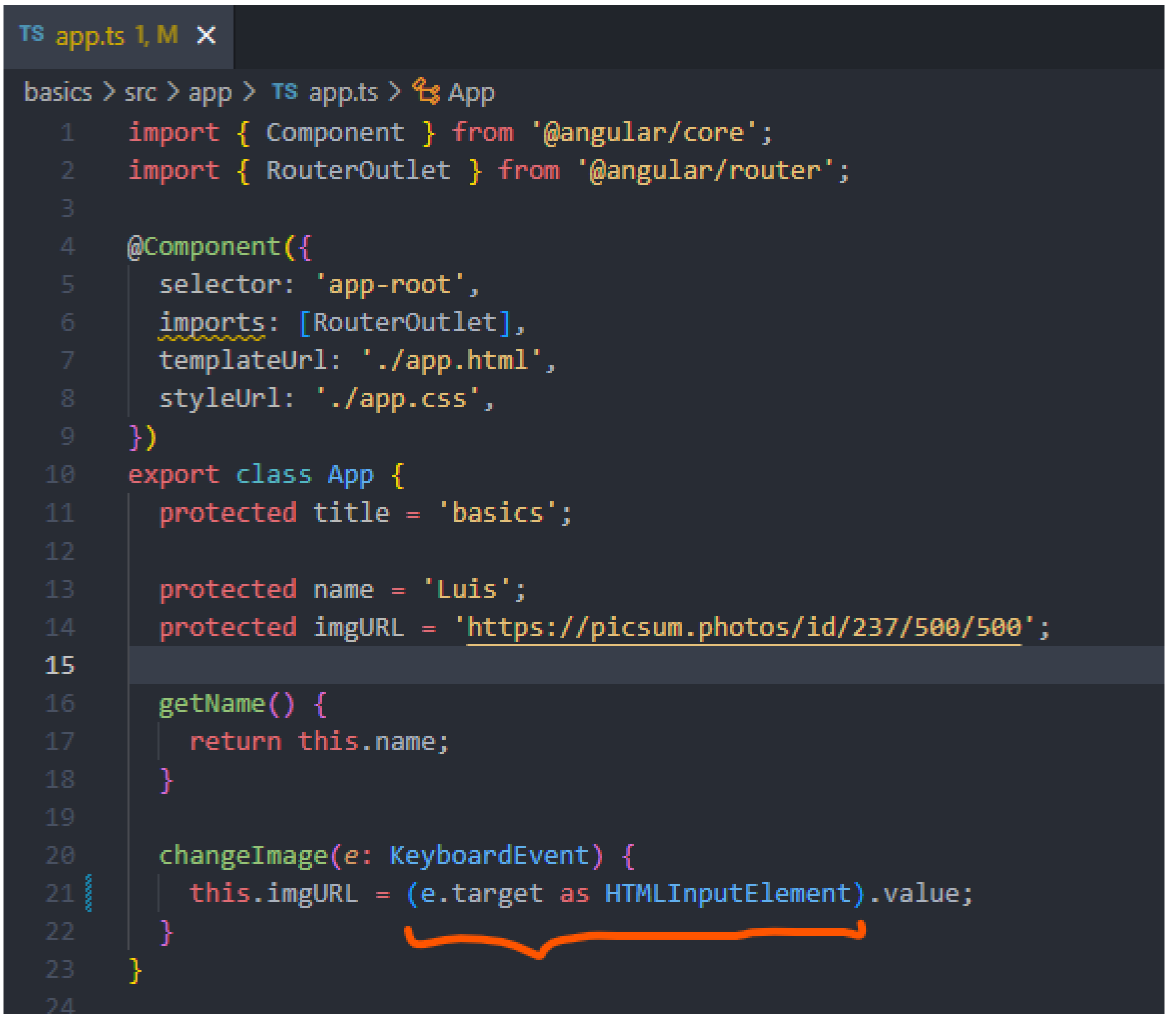
The **as keyword** will force the TypeScript compiler to assume a different type for an object or property. This behaviour is known as **TypeAssertion**. It's a way of helping the compiler understand the value. Technically, the event target interface is correct, but the HTMLInputElement interface is more accurate.

This interface (**HTMLInputElement**) should be applied to input elements. On this interface, the **value property** is defined after asserting the type:



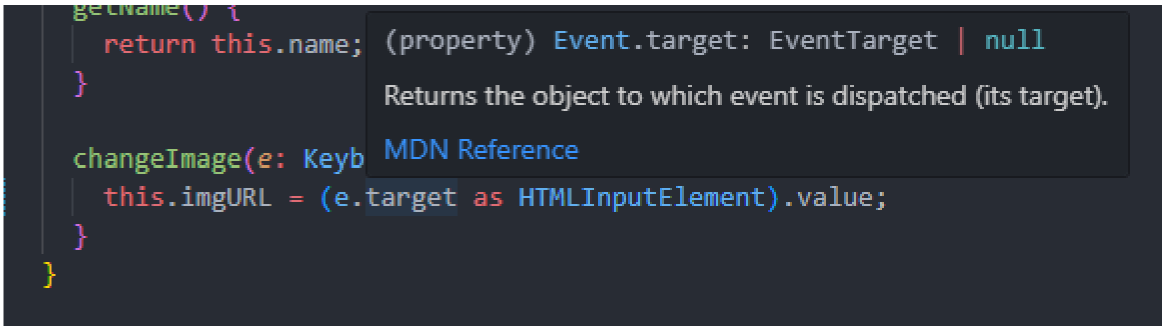
```
  }

changeImage(e: KeyboardEvent) {
  this.imgURL = (e.target as HTMLInputElement).value
  }
}
```

On my PC with latest version of Angular 20.0.5 – app.ts:



```
app.ts 1, M
basics > src > app > app.ts > App
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.html',
  styleUrl: './app.css',
})
export class App {
  protected title = 'basics';

  protected name = 'Luis';
  protected imgURL = 'https://picsum.photos/id/237/500/500';

  getName() {
    return this.name;
  }

  changeImage(e: KeyboardEvent) {
    this.imgURL = (e.target as HTMLInputElement).value;
  }
}
```

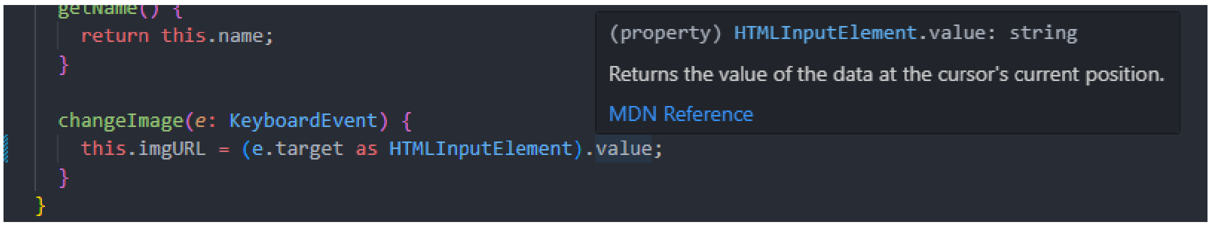
The error has gone away, If we hover the mouse over the **target property,** the type is being set to **EventTarget:**



```
    return this.name;
  }

  changeImage(e: Keyb
    this.imgURL = (e.target as HTMLInputElement).value;
  }
}
```

```
(property) Event.target: EventTarget | null
Returns the object to which event is dispatched (its target).
MDN Reference
```

Switching over to the **value property**, the type completely changes, the type has been changed to a **HTMLInputElement**, our new type has been asserted before accessing the **value**:

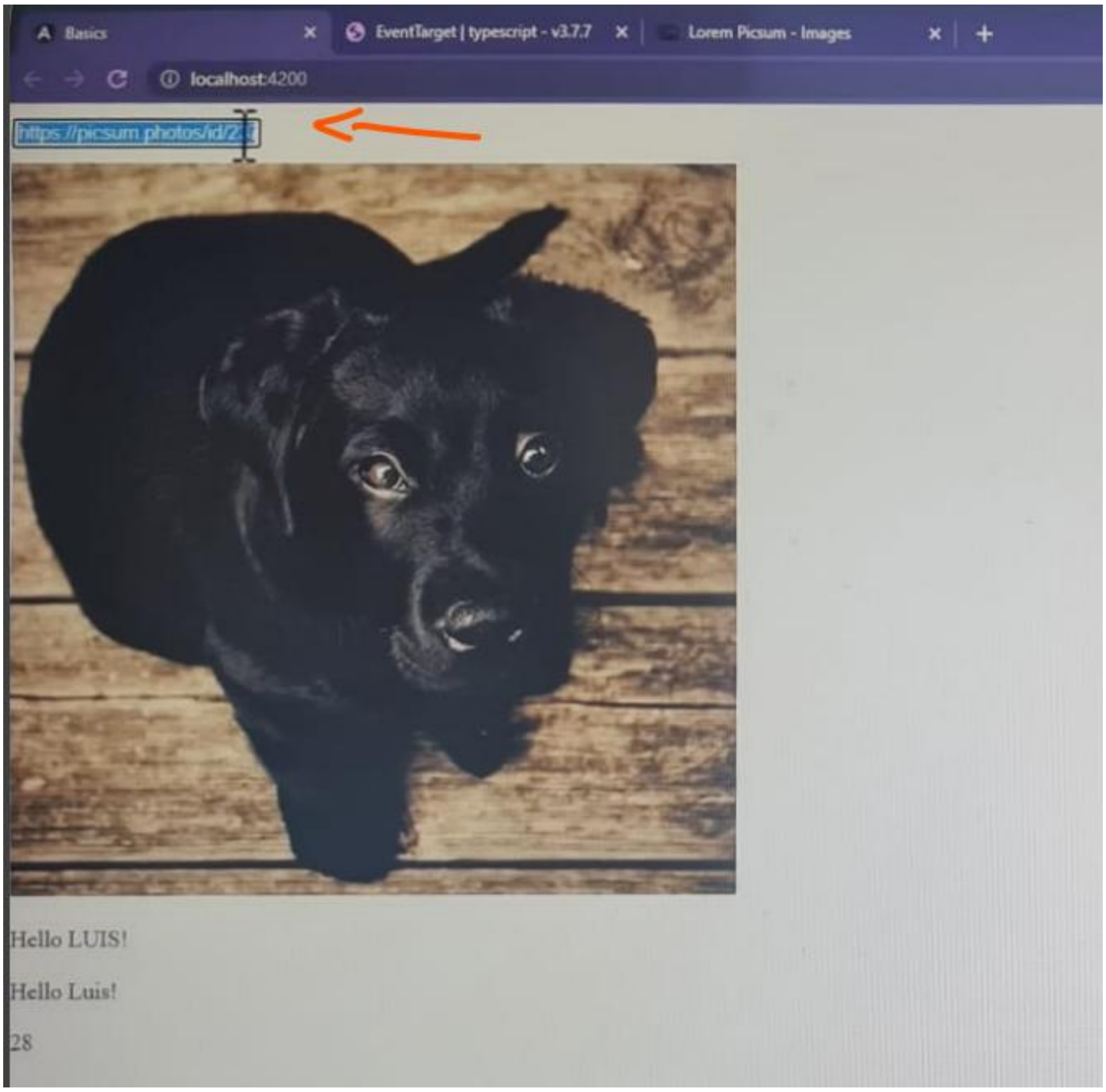


```
    return this.name;
  }

  changeImage(e: KeyboardEvent) {
    this.imgURL = (e.target as HTMLInputElement).value;
  }
}
```

```
(property) HTMLInputElement.value: string
Returns the value of the data at the cursor's current position.
MDN Reference
```

TypeScript is happy, let's check out the app in the browser:

The app is rendering at text input, it’s pretty filled with the property from our class, we can change the image to whatever we want, let’s change to image id/1025/..:



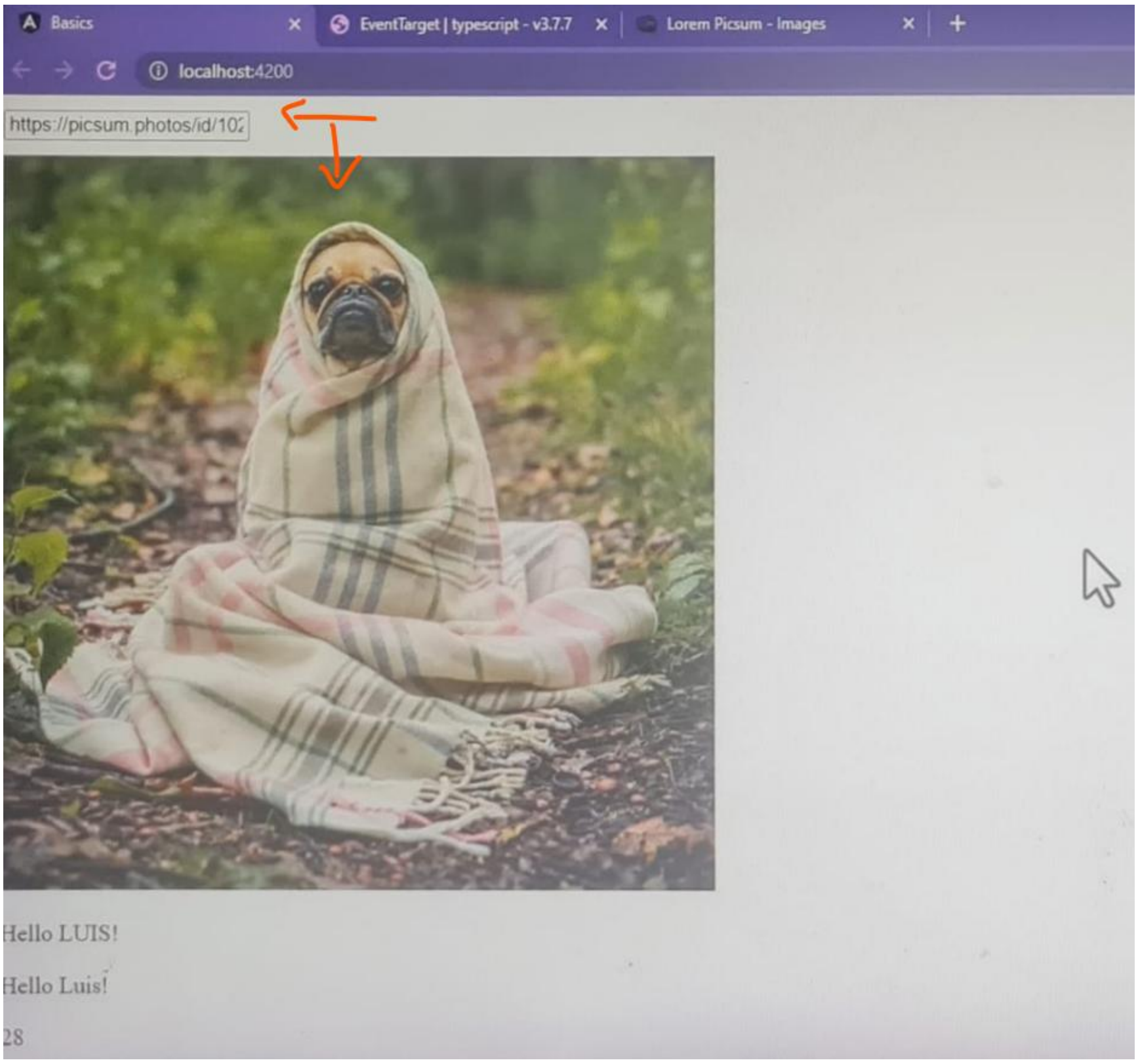
https://picsum.photos/id/2...

Hello LUIS!

Hello Luis!

28

We will get another picture of a dog.

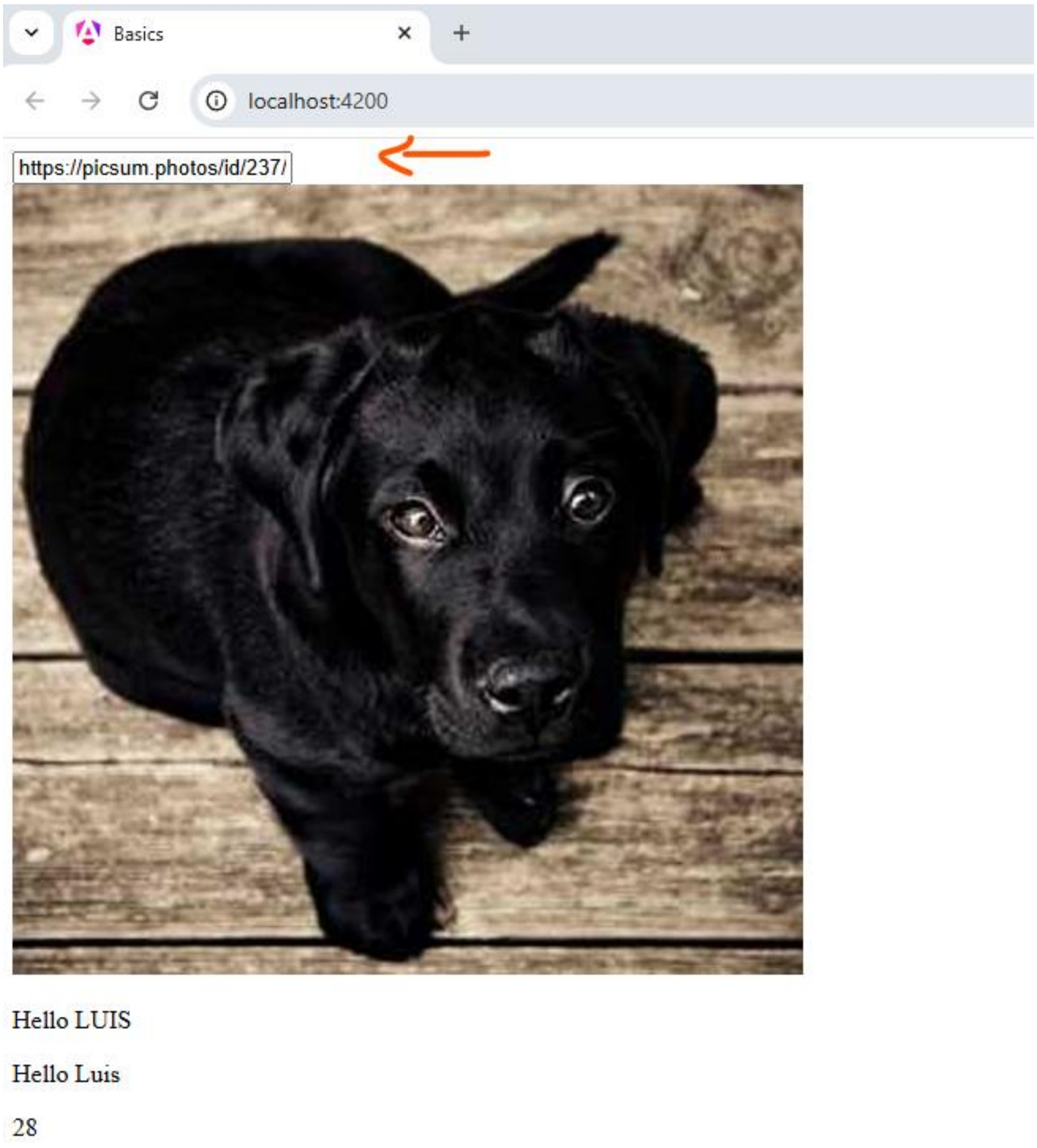


```
https://picsum.photos/id/102

Hello LUIS!

Hello Luis!

28
```

Our event is working as expected.

Event Binding is a powerful feature for listening to events on elements. In some cases, we may need to help the compiler understand the values data types by asserting the type.

On my PC with latest version of Angular 20.0.5 – Output:

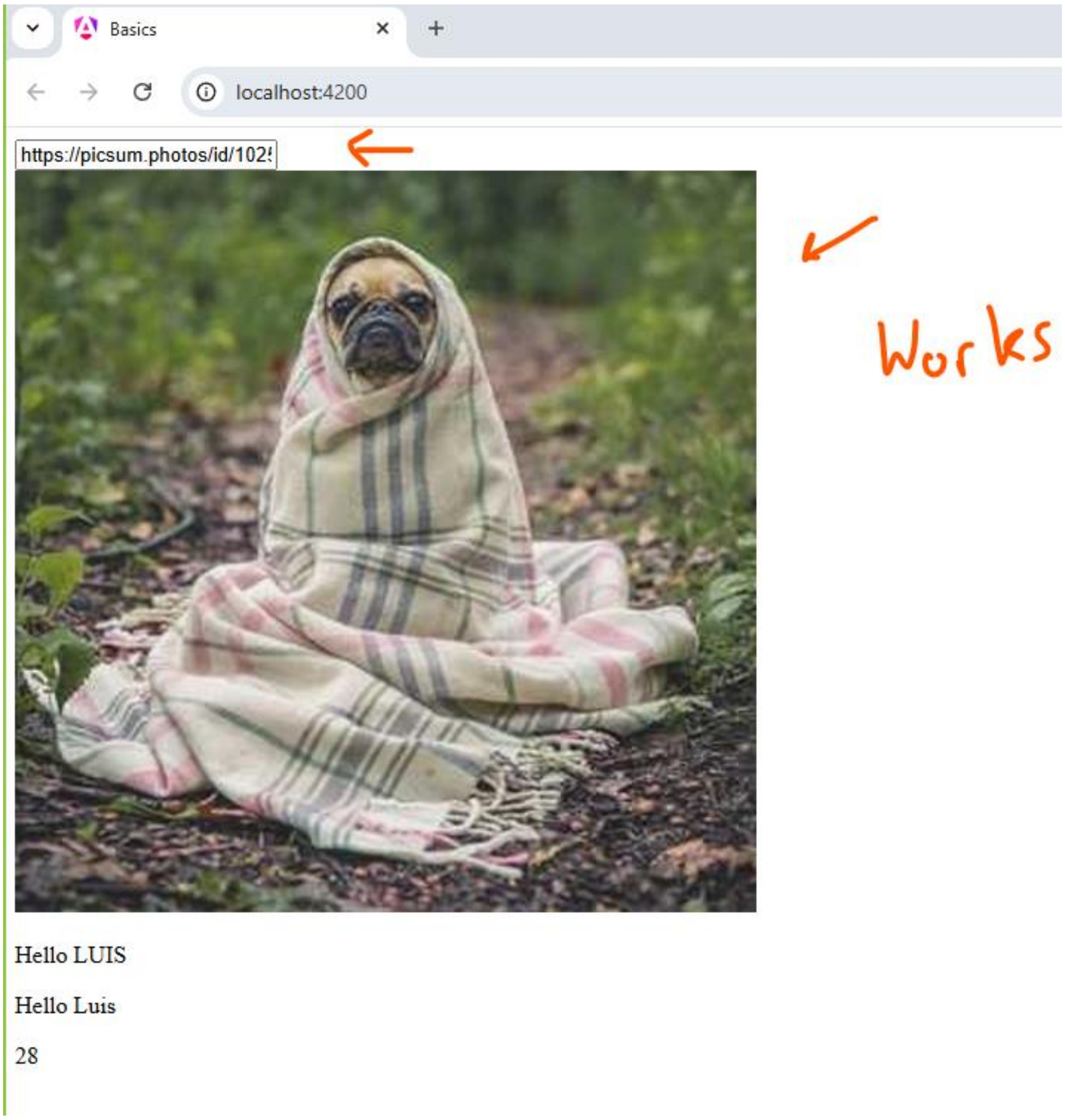


https://picsum.photos/id/237/

Hello LUIS

Hello Luis

28



https://picsum.photos/id/102!

Hello LUIS

Hello Luis

28