PESSIMISTIC DEVELOPMENT

YOU CAN'T WRITE PERFECT SOFTWARE

I WRITE PERFECT CODE.

– ME

HOW WE DRIVE

- WE LIKE TO THINK WE ARE THE BEST
- WE DRIVE DEFENSIVELY
- ANTICIPATE THE UNEXPECTED

WE CODE LIKE WE DRIVE...

WE BELIEVE WE ARE THE BEST DEVELOPER OUT THERE.

THIS IS SHOWN THROUGH DEFENSIVE CODING

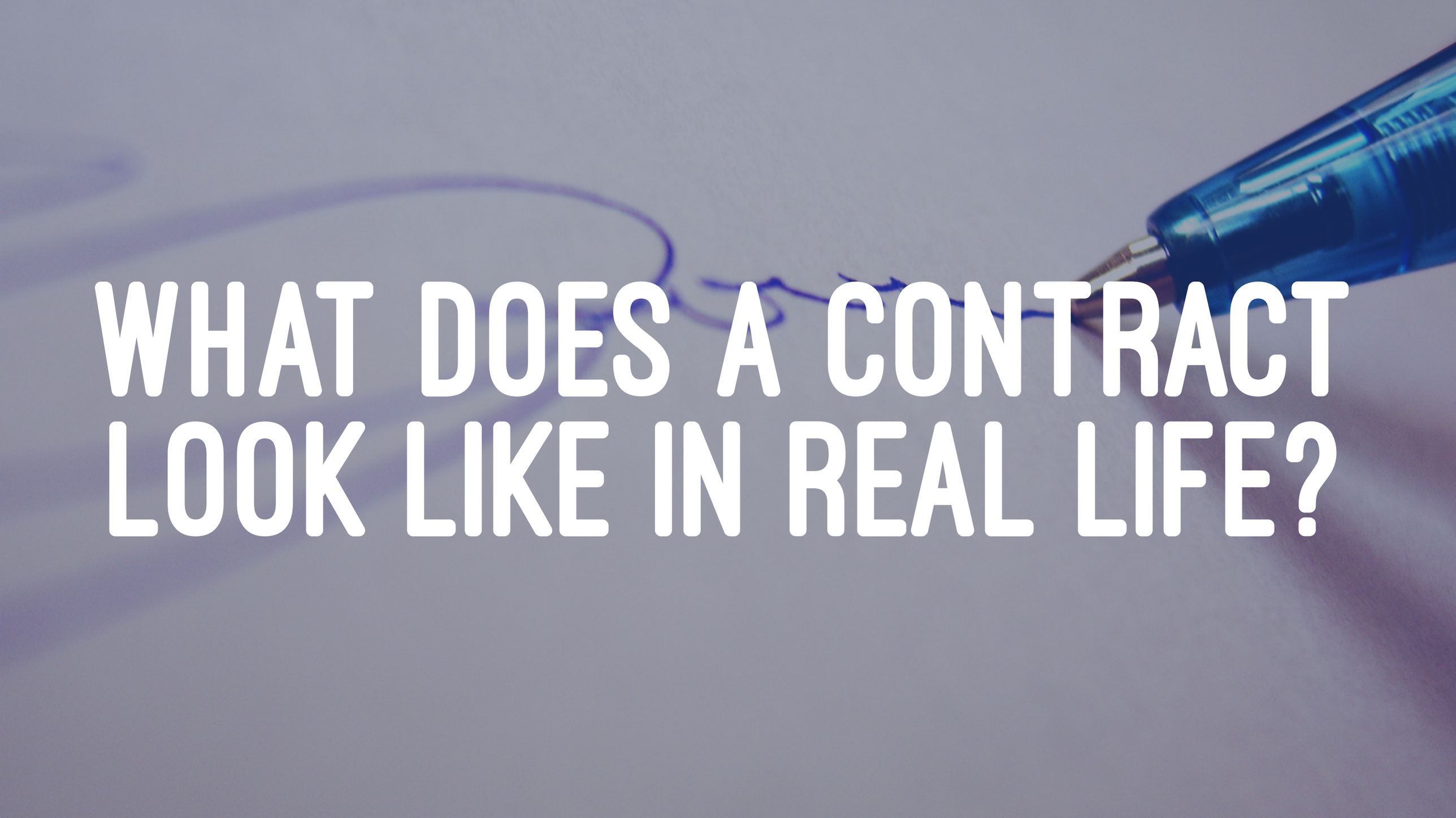
- WE VALIDATE
- USE ASSERTIONS
- CHECK FOR CONSISTENCY

BUT DON'T
TRUST
YOURSELF

BUILDING A PESSIMISTIC APPROACH

- DESIGN BY CONTRACT
- PROGRAM FOR THE IMPOSSIBLE
- ASSERTIVE PROGRAMMING
- ALERT GRACEFULLY

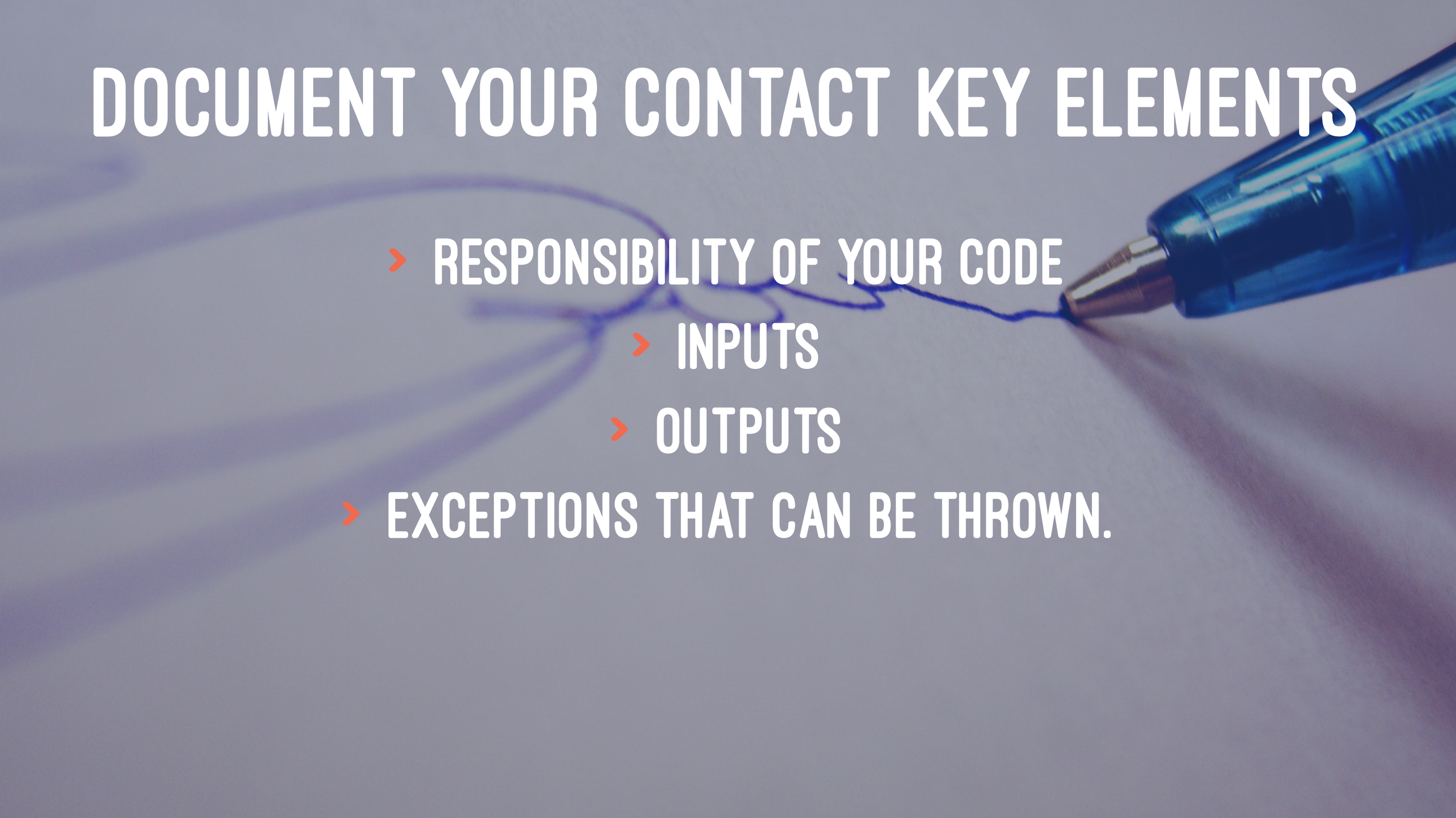

DESIGN BY
CONTRACT


WHAT DOES A CONTRACT
LOOK LIKE IN REAL LIFE?

WHAT DOES A CONTRACT LOOK LIKE IN CODE?

- DEFINES YOUR RIGHTS
- DEFINES RESPONSIBILITIES

DOCUMENT YOUR CONTACT KEY ELEMENTS



- RESPONSIBILITY OF YOUR CODE
- INPUTS
- OUTPUTS
- EXCEPTIONS THAT CAN BE THROWN.

CONTRACTS IN CODE

```
public interface IRepository<TEntityType>
{
    ApplicationUser CurrentUser { get; set; }

    IQueryable<TEntityType> GetAll();
    TEntityType Get(int id);
    IQueryable<TEntityType> FindBy(Expression<Func<TEntityType, bool>> predicate);
    TEntityType Add(TEntityType entity);
    void Delete(TEntityType entity);
    void Delete(int entity);
    TEntityType Edit(TEntityType entity);
    void Save();
}
```

PROGRAM DELIBERATELY

NOT BY COINCIDENCE

- ALWAYS BE AWARE OF WHAT YOU ARE DOING - STONE SOUP & BOILED FROGS
- DON'T CODE BLINDFOLDED
- PROCEED FROM A PLAN
- RELY ON RELIABLE THINGS -

PROGRAM FOR THE IMPOSSIBLE

NO ONE WOULD EVER DO THAT...

USERS AREN'T THE ONLY ONE WHO CREATE UNEXPECTED

THIS CODE WON'T BE USED 30 YEARS FROM NOW. SO TWO-DIGIT DATES ARE FINE

NO ONE WOULD PASS NULL TO US
SINCE IT COMES FROM THE DATABASE...

WE MAKE BIG ASSUMPTIONS THAT CAN LEAD TO TROUBLE.

EXPECT THE UNEXPECTED

DON'T

```
var something = obj.prop.name;
```

DO

```
var something;
if (obj && obj.prop && obj.prop.name)
{
  something = obj.prop.name;
}
```

ASSERTIVE PROGRAMMING

C# EXAMPLE

```
public ProductDto ConvertProductToDto(Product product)
 {
     if (product == null) throw new ArgumentNullException("product");


     ...

     return productDto;
 }
```

TESTING ASSERTIONS

```
[TestMethod]
[ExpectedException(typeof(ArgumentNullException),
    "A product of null was inappropriately allowed.")]
public void NullProductInConversion()
{
   var productDto = new ConvertProductToDto(null);
}
```

HANDLING ASSERTIONS WITH GRACE

- ALERT USER OF ERROR
- WHEN CALLING ANOTHER COMPONENT/SERVICE/MODULE. EXPECT THE UNEXPECTED
- BE PROACTIVE WITH ASSERTIONS IN YOUR OWN CODE

PRACTICAL USES

- ALWAYS ALERT ERRORS IN PROMISES ERROR BLOCK
- LOG API ERRORS TO SERVICES LIKE BUGSNAG (AND DEAL WITH THEM)