luv2code

Parameter Properties

Parameter Properties

Parameter Properties

- TypeScript offers a short-cut syntax for creating constructors

Parameter Properties

- TypeScript offers a short-cut syntax for creating constructors

- Helps to minimize the boilerplate code for constructors

Constructor - Traditional Approach

Constructor - Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }

    // accessors: get/set
    …
}
```

Constructor - Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }

    // accessors: get/set
    …
}
```

TypeScript offers a shortcut :-)

Constructor - Parameter Properties

Constructor - Parameter Properties

Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }
      // accessors: get/set
      …
}
```

Constructor - Parameter Properties

Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }
      // accessors: get/set
      …
}
```

The Short Cut

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

     // accessors: get/set
     …
}
```

Constructor - Parameter Properties

Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }
      // accessors: get/set
      …
}
```

The Short Cut

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

    // accessors: get/set
    …
```

Defines properties and assigns properties automagically.

Minimizes boilerplate coding!

Constructor - Parameter Properties

Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }
      // accessors: get/set
      …
}
```

The Short Cut

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

    // accessors: get/set
    …
```

Defines properties and assigns properties automagically.

Minimizes boilerplate coding!

Constructor - Parameter Properties

Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }
      // accessors: get/set
      …
}
```

The Short Cut

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

    // accessors: get/set
    …
```

Defines properties and assigns properties automagically.

Minimizes boilerplate coding!

Constructor - Parameter Properties

Traditional Approach

```
class Customer {

    private _firstName: string;
    private _lastName: string;

    constructor(theFirst: string, theLast: string) {
        this._firstName = theFirst;
        this._lastName = theLast;
    }
      // accessors: get/set
      …
}
```

The Short Cut

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

    // accessors: get/set
    …
```

Defines properties and assigns properties automagically.

Minimizes boilerplate coding!

Parameter Properties - In Action

Parameter Properties - In Action

File: Customer.ts

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

    // accessors: get/set
    …
}
```

Parameter Properties - In Action

File: Customer.ts

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

    // accessors: get/set
    …
}
```

Defines properties and assigns properties automagically.

Minimizes boilerplate coding!

Parameter Properties - In Action

File: Customer.ts

```
class Customer {

    constructor(private _firstName: string,
                private _lastName: string) {

    }

    // accessors: get/set
    …
}
```

```
// now let's use it
let myCustomer = new Customer("Martin", "Dixon");

myCustomer.firstName = "Susan";
console.log(myCustomer.firstName);
```