

Functions



Brice Wilson

@brice_wilson www.BriceWilson.net



Overview



Functions in TypeScript versus JavaScript

Parameter types and return types

Arrow functions

Function types

Parameters

- Optional parameters
- Default parameters
- Rest parameters

Overloaded functions



Functions in TypeScript Versus JavaScript

TypeScript

Types (of course!)

Arrow functions

Function types

Required and optional parameters

Default parameters

Rest parameters

Overloaded functions

JavaScript

No types

Arrow functions (ES2015)

No function types

All parameters are optional


Default parameters (ES2015)

Rest parameters (ES2015)

No overloaded functions




Parameter Types and Return Types



```
function CreateCustomerID(name: string, id: number): string {  
    return name + id;  
}
```





```
let arr = allBooks.filter(function(book) {  
    return book.author === 'Herman Melville';  
});
```

```
let arr = allBooks.filter(book => book.author === 'Herman Melville');
```

Arrow Functions

Concise syntax for anonymous functions

“this” is captured at function creation – not invocation



Arrow Function Syntax

```
myBooks.forEach(() => console.log('Done reading!'));
```



Arrow Function Syntax

```
myBooks.forEach(() => console.log('Done reading!'));
```

```
myBooks.forEach(title => console.log(title));
```



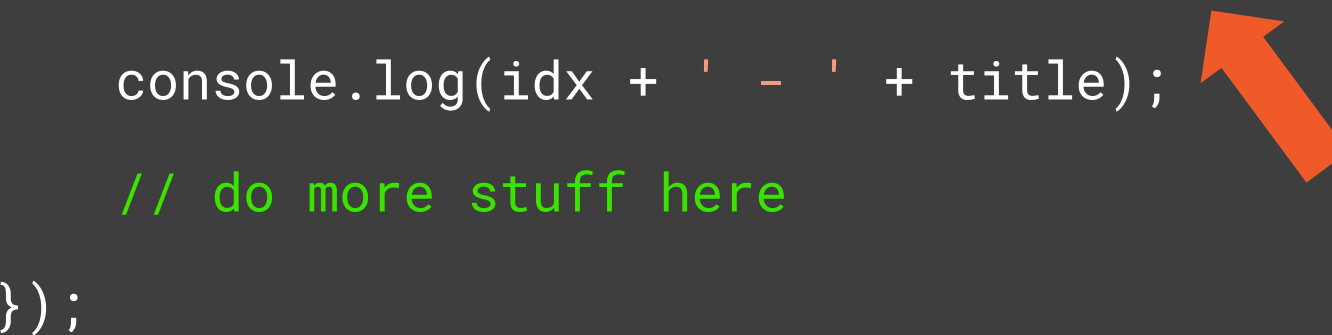
Arrow Function Syntax

```
myBooks.forEach(() => console.log('Done reading!'));  
myBooks.forEach(title => console.log(title));  
myBooks.forEach((title, idx, arr) => console.log(idx + ' - ' + title));
```



Arrow Function Syntax

```
myBooks.forEach(() => console.log('Done reading!'));  
myBooks.forEach(title => console.log(title));  
myBooks.forEach((title, idx, arr) => console.log(idx + ' - ' + title));  
myBooks.forEach((title, idx, arr) => {  
    console.log(idx + ' - ' + title);  
    // do more stuff here  
});
```



Capturing “this” in JavaScript

```
function Book() {  
  let self = this;   
  self.publishDate = 2016;   
  setInterval(function() {  
    console.log(self.publishDate);   
  }, 1000);  
}
```

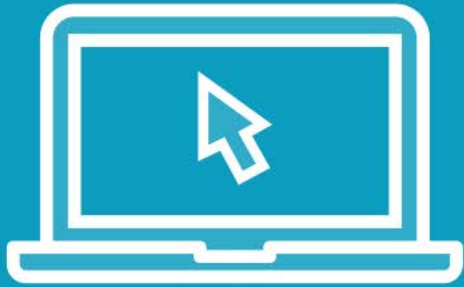


Capturing “this” in Arrow Functions

```
function Book() {  
  this.publishDate = 2016;   
  setInterval(() => {  
    console.log(this.publishDate);   
  }, 1000)  
}
```



Demo



Using arrow functions



```
function PublicationMessage(year: number): string {  
    return 'Date published: ' + year;  
}  
  
let publishFunc: (someYear: number) => string;
```



Function Types


Combination of parameter types and return type

Variables may be declared with function types

Function assigned must have the same signature as the variable type



```
function PublicationMessage(year: number): string {  
    return 'Date published: ' + year;  
}
```



```
let publishFunc: (someYear: number) => string;
```

```
publishFunc = PublicationMessage;
```

```
let message: string = publishFunc(2016);
```

Function Types

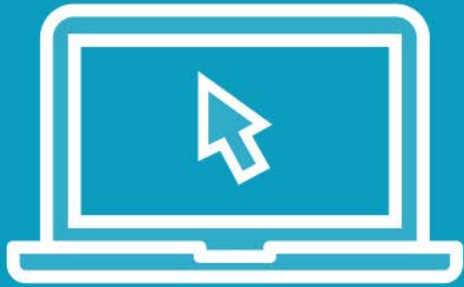
Combination of parameter types and return type

Variables may be declared with function types

Function assigned must have the same signature as the variable type



Demo



Declaring and using function types



```
function CreateCustomer(name: string, age?: number) { }
```



Optional and Default Parameters

Optional parameters denoted with “?” after parameter name




```
function CreateCustomer(name: string, age?: number) { }
```

```
function GetBookByTitle(title: string = 'The C Programming Language') { }
```



Optional and Default Parameters

Optional parameters denoted with “?” after parameter name

Must appear after all required parameters

Default parameters may be set to a literal value or an expression



```
function CreateCustomer(name: string, age?: number) { }
```

```
function GetBookByTitle(title: string = 'The C Programming Language') { }
```

```
function GetBookByTitle(title: string = GetMostPopularBook()) { }
```

Optional and Default Parameters

Optional parameters denoted with “?” after parameter name

Must appear after all required parameters

Default parameters may be set to a literal value or an expression



```
function GetBooksReadForCust(name: string, ...bookIDs: number[]) { }
```



Rest Parameters

Collects a group of parameters into a single array

Denoted with an ellipsis prefix on last parameter



```
function GetBooksReadForCust(name: string, ...bookIDs: number[]) { }
```

```
let books = GetBooksReadForCust('Leigh', 2, 5);
```



Rest Parameters

Collects a group of parameters into a single array

Denoted with an ellipsis prefix on last parameter



```
function GetBooksReadForCust(name: string, ...bookIDs: number[]) { }
```

```
let books = GetBooksReadForCust('Leigh', 2, 5);
```

```
let books = GetBooksReadForCust('Daniel', 2, 5, 12, 42);
```



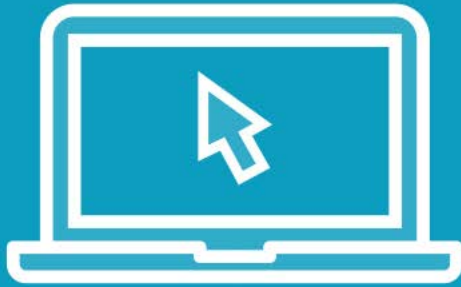
Rest Parameters

Collects a group of parameters into a single array

Denoted with an ellipsis prefix on last parameter



Demo



Declaring parameters

- Optional
- Default
- Rest



Function Overloads

One symbol name

Multiple function
types

One
Implementation
with type guards



Implementing Function Overloads

```
function GetTitles(author: string): string[];
```



Implementing Function Overloads

```
function GetTitles(author: string): string[];  
function GetTitles(available: boolean): string[];
```





Implementing Function Overloads

```
function GetTitles(author: string): string[];  
function GetTitles(available: boolean): string[];  
function GetTitles(bookProperty: any): string[] {
```

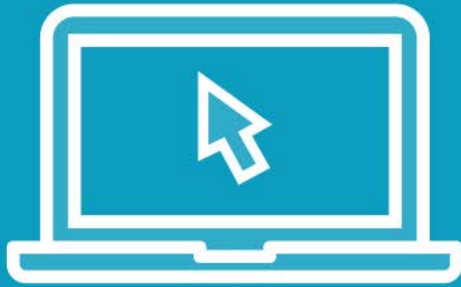


Implementing Function Overloads

```
function GetTitles(author: string): string[];
function GetTitles(available: boolean): string[];
function GetTitles(bookProperty: any): string[] {
    if(typeof bookProperty == 'string') { 
        // get books by author, add to foundTitles
    }
    else if(typeof bookProperty == 'boolean') { 
        // get books by availability, add to foundTitles
    }
    return foundTitles;
}
```



Demo



Overloading functions



Summary



JavaScript functions with more features!!!

Arrow functions

Function types

Parameters

Overloads

