SOEN 287

Chapter 4: JavaScript (4) Supplement: Arrow Function

Dr. Yuhong Yan
CSE, Concordia University
Winter, 2024

Arrow Function (1) – no parameters

- Returns a fixed value

```
const greet = () =>{
   return "Hello, World!";
};


Console.log(greet()); //output Hello, World!
```

Arrow Function (2) – single parameters

- parentheses around the parameter are optional

```
const double = number => {
  return number * 2;
};

console.log(double(4)); // Output: 8
```

Arrow Function (3) – multiple parameters

- Requires parentheses around the parameter

```
const add = (a, b) => {
  return a + b;
};

console.log(add(5, 3)); // Output: 8
```

Arrow Function (4) – Single Line Body

- If the function body consists of a single statement, you can omit the curly braces and the return keyword. The result of the expression will be returned automatically.

```
const square = x => x * x;

console.log(square(5)); // Output: 25
```

Arrow Function (5) – **Returning Object Literals**

- When returning an object literal, enclose the literal in parentheses to distinguish it from the function's body.

```
const createPerson = (name, age) => ({
  name: name,
  age: age
});

console.log(createPerson("Alice", 30)); //
Output: { name: 'Alice', age: 30 }
```

Arrow Function (5) – **Returning Object Literals**

- When returning an object literal, enclose the literal in parentheses to distinguish it from the function's body.

```
const createPerson = (name, age) => ({
  name: name,
  age: age
});

console.log(createPerson("Alice", 30)); //
Output: { name: 'Alice', age: 30 }
```

Arrow Function (6) – **As Callback Functions**

- Arrow functions are commonly used as callbacks for methods like map, filter, reduce, etc.

```
const numbers = [1, 2, 3, 4];
const squares = numbers.map(x => x * x);

console.log(squares); // Output: [1, 4, 9,
16]
```

Arrow Function (7) – **No Binding of this**

- Arrow functions do not bind their own this. They inherit this from the parent scope at the time they are defined. This is particularly useful for event handlers and callbacks.

```
class Button {
  constructor(label) {
    this.label = label;
    document.addEventListener('click', () =>
{
      console.log(this.label);
    });
  }
}

const myButton = new Button('Click Me!');
// When the document is clicked, "Click Me!"
will be logged to the console.
```

The End