**<h3>What is Arrow Function</h3>**

<p>Arrow function might be one of the most impressive new features in the JavaScript ES6 syntax, it offers us a new way to code a function, which omits the “function” keyword but use an arrow =>. In most cases, there is no difference between using the arrow function and the normal javascript function, developers can just choose the way they like. However, of course there IS some differences between these two ways to write functions.</p>

<br/>

**<h3>The difference is “this”!</h3>**

<p>The most possible case when meeting a bug which is related to the wrong usage of arrow functions is probably due to the misunderstanding of what is “this” in an arrow function. (Especially when programming with React.js, some people may code a “onClick” function on the &lt;button&gt; tag inside the JSX, such as <strong>&lt;button onClick={this.abc} /&gt;</strong>, and then inside this “abc” function, the keyword “this” is used, then probably some error will be thrown up, but if “abc” is an arrow function, then everything will be good.) Weather program with React.js or not, this special feature about arrow function is that “Arrow function does not define its own <strong>this</strong> value”. Conversely, a normal JavaScript function will define its own “this”.</p>

<hr />

<p><strong>

function Person() {<br/>

&emsp;&emsp;this.name = “Alice”;<br/>

&emsp;&emsp;setInterval(function abc() {<br/>

&emsp;&emsp;&emsp;&emsp;console.log(this.name);<br/>

&emsp;&emsp;});<br/>

}</strong>

<hr/>

<strong>

function Person() {<br/>

&emsp;&emsp;this.name = “Alice”;<br/>

&emsp;&emsp;setInterval( () =&gt; {<br/>

&emsp;&emsp;&emsp;&emsp;console.log(this.name);<br/>

&emsp;&emsp;});<br/>

}

</strong></p>

<hr/>

<p>These two Person functions are declared by the normal “function” keyword, so they will define the “this” value when they get initiated like <strong>var p1 = new Person();</strong>. Then they are using different ways to declare a callback function for “setInterval()”. In the former version, a normal function “abc” is declared and the “this” inside the “abc” will be the Global Object rather than the “this” we want (which should point to the instance of Person). In the latter version, “abc” is declared as an Arrow function, things will be different, the “this” value will not be overwritten, so “this.name” will still be Alice.</p>

<p>Due to this feature of arrow function, you can see that it might not be a good idea to use arrow function to define a method for an object like this.</p>

<hr/>

<p><strong>

var person = {<br/>

&emsp;&emsp;name: “Alice”,<br/>

&emsp;&emsp;getName1: () =&gt; this.name, <br/>

&emsp;&emsp;getName2: function () { return this.name; }<br/>

}</strong></p>

<hr/>

<p>If try the above code out in the console, then will find that the <strong>person.getName1();</strong> will generate nothing, and if you change the “this.name” to “this”, probably the “Window” object will be printed out. However, <strong>person.getName2();</strong> will work fine just as it supposed to. It is obviously that the arrow function did not overwrite the “this” value, so the “this” used inside it is same with the “this” in the parent block, which probably is the “Window” object if this is “person” variable is declared as global variable in the console of the browser. On the other hand, “getName2” define its own “this”, which will be the one who invoke it, and that is variable “person” in this case.</p>

<br/>

**<h3>There are other differences as well</h3>**

<p>From my perspective, the meaning of “this” keyword is the most different point between normal function and arrow function. Yet there are still other differences as well, such as the arrow function does not have their own “arguments” objects as well, while the normal function does have.</p>

<p>And also, in the past, before we have keyword “class” in the JavaScript. We use function as constructor to make something works similar with “class” from OOP languages. An example will be the function “Person” I provided above. The arrow function can not be used here as constructor, which will throw an error if you try to use <strong>var p3 = Person();</strong> while the Person() is defined by an arrow function.</p>