

ES6 101

let, const, arrow functions, template strings

What is ES2015 (aka ES6)?

- Latest version of ECMAScript (JS standard) specification
- Finalized in 2015

Features

- Arrow functions
- Classes
- Template and multi-line strings
- Destructuring assignment and spread
- Enhanced object literals
- Promises
- let and const
- modules
- for of loop
- generators - separate lecture
- lots more

Browser Support

| | | Compilers/polyfills | | | | | | Desktop browsers | | | | | | | | | |
|--|---|---------------------|---------|--------------------------------|---------|-----------------------|----------|------------------------|-------|------------------------|------------------------|------------------------|-----------|-------|------------|--------------|---------------|
| | | 97% | 56% | 71% | 48% | 59% | 18% | 5% | 11% | 83% | 93% | 95% | 86% | 92% | 94% | 94% | 96% |
| Feature name | | Current browser | Traceur | Babel + core-js ^[2] | Closure | Type-Script + core-js | es6-shim | KQ 4.14 ^[3] | IE 11 | Edge 13 ^[4] | Edge 14 ^[4] | Edge 15 ^[4] | FF 45 ESR | FF 50 | FF 51 Beta | FF 52 Aurora | FF 53 Nightly |
| ● RegExp "y" and "u" flags | ▶ | 5/5 | 3/5 | 3/5 | 0/5 | 0/5 | 0/5 | 0/5 | 0/5 | 5/5 | 5/5 | 5/5 | 2/5 | 5/5 | 5/5 | 5/5 | 5/5 |
| ● destructuring.declarations | ▶ | 22/22 | 20/22 | 21/22 | 18/22 | 15/22 | 0/22 | 0/22 | 0/22 | 0/22 | 21/22 | 22/22 | 19/22 | 21/22 | 21/22 | 21/22 | 21/22 |
| ● destructuring.assignment | ▶ | 24/24 | 23/24 | 24/24 | 16/24 | 19/24 | 0/24 | 0/24 | 0/24 | 0/24 | 23/24 | 24/24 | 21/24 | 23/24 | 23/24 | 23/24 | 23/24 |
| ● destructuring.parameters | ▶ | 23/23 | 19/23 | 20/23 | 17/23 | 15/23 | 0/23 | 0/23 | 0/23 | 0/23 | 22/23 | 23/23 | 18/23 | 19/23 | 20/23 | 20/23 | 22/23 |
| ● Unicode code point escapes | ▶ | 2/2 | 1/2 | 1/2 | 1/2 | 1/2 | 0/2 | 0/2 | 0/2 | 2/2 | 2/2 | 2/2 | 1/2 | 1/2 | 1/2 | 1/2 | 2/2 |
| ● new.target | ▶ | 2/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 0/2 | 1/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 | 2/2 |
| Bindings | | | | | | | | | | | | | | | | | |
| ● const | ▶ | 16/16 | 14/16 | 14/16 | 14/16 | 14/16 | 0/16 | 2/16 | 12/16 | 12/16 | 16/16 | 16/16 | 12/16 | 12/16 | 16/16 | 16/16 | 16/16 |
| ● let | ▶ | 12/12 | 10/12 | 10/12 | 10/12 | 10/12 | 0/12 | 0/12 | 10/12 | 10/12 | 12/12 | 12/12 | 10/12 | 10/12 | 12/12 | 12/12 | 12/12 |
| ● block-level function declaration ^[12] | Ⓢ | Yes | Yes | Yes | Yes | No | No | No | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes |
| Functions | | | | | | | | | | | | | | | | | |
| ● arrow functions | ▶ | 13/13 | 11/13 | 9/13 | 10/13 | 9/13 | 0/13 | 0/13 | 0/13 | 13/13 | 13/13 | 13/13 | 13/13 | 13/13 | 13/13 | 13/13 | 13/13 |
| ● class | ▶ | 24/24 | 17/24 | 19/24 | 13/24 | 19/24 | 0/24 | 0/24 | 0/24 | 24/24 | 24/24 | 24/24 | 24/24 | 24/24 | 24/24 | 24/24 | 24/24 |
| ● super | ▶ | 8/8 | 7/8 | 4/8 | 5/8 | 7/8 | 0/8 | 0/8 | 0/8 | 8/8 | 8/8 | 8/8 | 8/8 | 8/8 | 8/8 | 8/8 | 8/8 |
| ● generators | ▶ | 27/27 | 24/27 | 24/27 | 16/27 | 0/27 | 0/27 | 0/27 | 0/27 | 27/27 | 27/27 | 27/27 | 25/27 | 25/27 | 25/27 | 25/27 | 25/27 |
| Built-ins | | | | | | | | | | | | | | | | | |
| ● typed arrays | ▶ | 46/46 | 0/46 | 45/46 | 0/46 | 45/46 | 0/46 | 8/46 | 16/46 | 44/46 | 46/46 | 46/46 | 42/46 | 45/46 | 45/46 | 46/46 | 46/46 |
| ● Map | ▶ | 19/19 | 14/19 | 19/19 | 14/19 | 19/19 | 15/19 | 0/19 | 8/19 | 18/19 | 18/19 | 19/19 | 18/19 | 18/19 | 18/19 | 18/19 | 19/19 |
| ● Set | ▶ | 19/19 | 14/19 | 19/19 | 14/19 | 19/19 | 15/19 | 0/19 | 8/19 | 18/19 | 18/19 | 19/19 | 18/19 | 18/19 | 18/19 | 18/19 | 19/19 |
| ● WeakMap | ▶ | 12/12 | 6/12 | 12/12 | 9/12 | 12/12 | 0/12 | 0/12 | 6/12 | 11/12 | 11/12 | 12/12 | 10/12 | 11/12 | 11/12 | 11/12 | 12/12 |
| ● WeakSet | ▶ | 11/11 | 5/11 | 11/11 | 8/11 | 11/11 | 0/11 | 0/11 | 0/11 | 10/11 | 10/11 | 11/11 | 9/11 | 10/11 | 10/11 | 10/11 | 11/11 |

to the rescue

Babel translates all the ES2015 fancy features to “classic” JavaScript so that even oldie browsers can run it.

Also a good tool for learning ES2015 features

let

~~var~~

let

Difference between var and let

- var is function scoped
- let is block scoped

block scoped

```
for (let i = 0; i < 10; i++) {  
  
}  
  
console.log(i);
```

✖ ▶ Uncaught ReferenceError: i is not defined
at <anonymous>:7:15


What kind of blocks?

- function blocks
- if blocks
- for loop blocks
- switch blocks
- even plain blocks


const

const

- Like let, except you can't change the value of the variable
- For values that should not change
- Commonly used for
 - constants, like PI, or string constants
 - names you used for modules
 - pure functional programming



```
const PI = 3.14159265  
35897932384626433832795028841971693993751;
```



```
const express = require('express');  
const Promise = require('bluebird');
```

```
const express = require('express');  
const Promise = require('bluebird');
```

```
const PI = 3.14159265  
35897932384626433832795028841971693993751;
```

Arrow Functions

Arrow function w one parameter

```
1 let times2 = x => x * 2;
```

```
2
```

```
3
```

```
4
```

```
1 "use strict";
```

```
2
```

```
3 var times2 = function times2(x) {
```

```
4     return x * 2;
```

```
5
```

```
};
```

Arrow function w multiple parameters

```
1 let add = (x, y) => x + y;
```

```
2
```

```
3
```

```
4
```

```
1 "use strict";
```

```
2
```

```
3 var add = function add(x, y) {
```

```
4     return x + y;
```

```
5 };
```


Arrow functions with multiple lines

```
1 let add = (x, y) => {  
2   return x + y;  
3 };  
4  
5
```

```
1 "use strict";  
2  
3 var add = function add(x, y) {  
4   return x + y;  
5 };
```

Arrow functions are different than normal functions

- They are not constructors - cannot *new* them
- They do now define a new *this* variable within them, this within an arrow function will be the outer non-arrow function's *this*

Arrow functions are not constructors

```
> let Cat = (name) => {  
  this.name = name;  
};
```

```
let garfield = new Cat('Garfield');
```

✖ ▶ Uncaught TypeError: Cat is not a constructor(...)

Arrow functions as inner functions

```
1 class Person {  
2   constructor(name) {  
3     this.name = name;  
4   }  
5  
6   greet() {  
7     console.log(`${this.name} is getting ready...`);  
8     setTimeout(function() {  
9       console.log(`Hello! My name is ${this.name}.`);  
10    }, 1000);  
11   }  
12 }  
13  
14  
15 let bob = new Person('Bob');  
16 bob.greet();
```

Arrow functions as inner functions

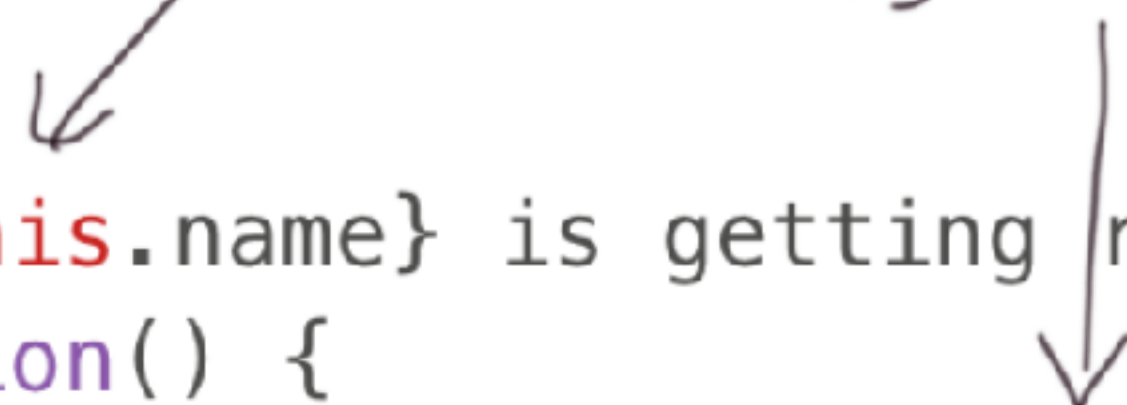
```
1 class Person {  
2   constructor(name) {  
3     this.name = name;  
4   }  
5  
6   greet() {  
7     console.log(`${this.name} is getting ready...`);  
8     setTimeout(function() {  
9       console.log(`Hello! My name is ${this.name}.`);  
10    }, 1000);  
11   }  
12 }  
13  
14  
15 let bob = new Person('Bob');  
16 bob.greet();
```

"Bob is getting ready..."
"Hello! My name is ."

Arrow functions as inner functions

```
1 class Person {  
2   constructor(name) {  
3     this.name = name;  
4   }  
5  
6   greet() {  
7     console.log(`${this.name} is getting ready...`);  
8     setTimeout(function() {  
9       console.log(`Hello! My name is ${this.name}.`);  
10    }, 1000);  
11   }  
12 }
```

this this is not
the same this as
this this



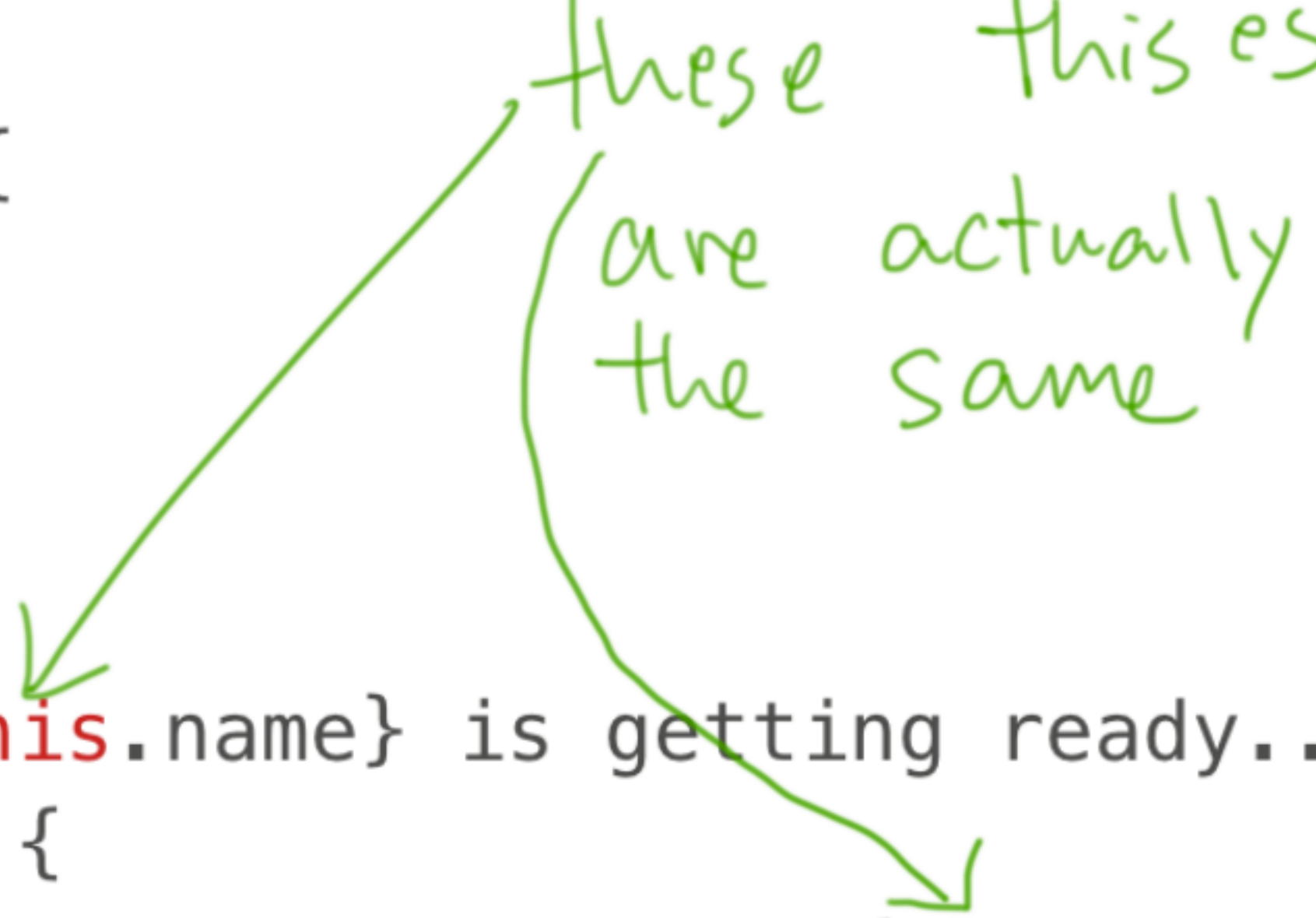
```
15 let bob = new Person('Bob');  
16 bob.greet();
```

"Bob is getting ready..
"Hello! My name is ."

Arrow functions as inner functions

```
1 class Person {  
2   constructor(name) {  
3     this.name = name;  
4   }  
5  
6   greet() {  
7     console.log(`${this.name} is getting ready...`);  
8     setTimeout(() => {  
9       console.log(`Hello! My name is ${this.name}.`);  
10    }, 1000);  
11   }  
12 }
```

these *this*es
are actually
the same



"Bob is getting ready..."
"Hello! My name is Bob."

Template Strings

Template Strings

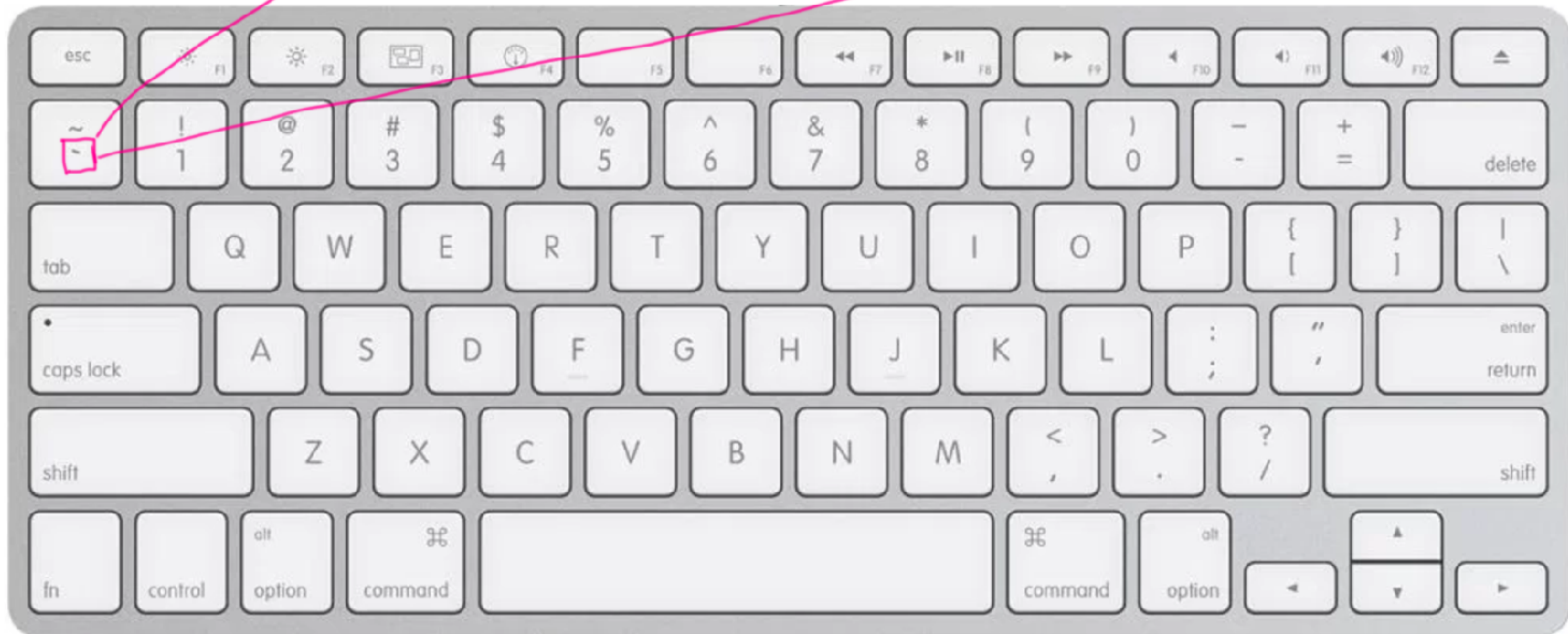
```
let string = `Hello, world!`;  
console.log(string);
```

Template Strings

```
let string = `Hello, world!`;
```

```
console.log(string);
```

UP TICK S



Template string substitutions

```
let subject = 'world';  
let string = `Hello, ${subject}!`;  
  
console.log(string);
```

Template string substitutions

```
let subject = 'world';  
let string = `Hello, ${subject}!`;  
  
console.log(string);
```

Multi-line strings

```
let haiku = `  
round lumps of cells grow  
up to love porridge later  
become The Supremes  
`;
```

Multi-line strings

```
let title = 'Welcome to my corner of the WWW';  
let html = `  
<h1>${title}</h1>  
<p>Hello, ${name}! Take a look around.</p>  
`;  
;
```