

Blockchain for Industrial Engineers: Decentralized Application Development

**บล็อกเชนสำหรับวิศวกรอุตสาหกรรม: การพัฒนาแอปพลิเคชันแบบ
กระจายศูนย์**

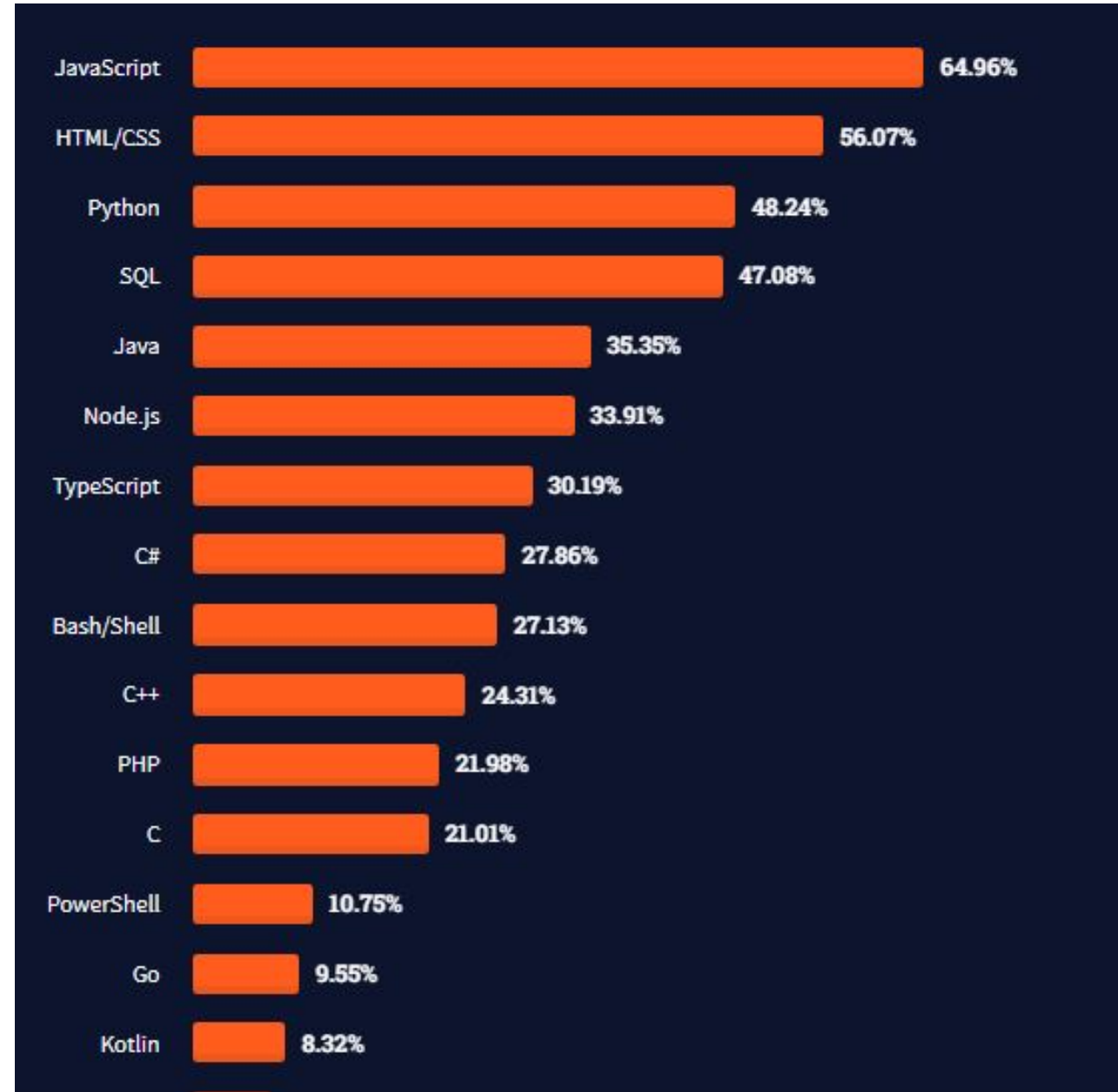
```
177 $total: $total + $taxes;
178 $total: $total + $taxes;
179 });
180 $taxes = $taxes;
181 $taxes = array();
182 $taxes = array();
183 $sort_order = array();
184 $results = $this->model->extension->getExtensions('total');
185
186 foreach ($results as $key => $value) {
187     if (isset($value['code'])) {
188         $code = $value['code'];
189     } else {
190         $code = $value['key'];
191     }
192     $sort_order[$key] = $this->config->get($code - 'sort_order');
193 }
194
195 $sort_order[$key] = $this->config->get($code - 'sort_order');
196
197 array_multisort($sort_order, SORT_ASC, $results);
198
199 foreach ($results as $result) {
200     if (isset($result['code'])) {
201         $code = $result['code'];
202     } else {
203         $code = $result['key'];
204     }
205     $code = $result['key'];
206 }
```

```
354 Carousel.prototype.get
355 this.$items = item.parent().children()
356 return this.$items.index(item || this.$active)
357 }
358
359 Carousel.prototype.getItemForDirection = function (direction, active) {
360     var delta = direction == 'prev' ? -1 : 1
361     var activeIndex = this.getItemIndex(active)
362     var itemIndex = (activeIndex + delta) % this.$items.length
363     return this.$items.eq(itemIndex)
364 }
365
366 Carousel.prototype.to = function (pos) {
367     var that = this
368     var activeIndex = this.getItemIndex(this.$active = this.$element.find('.item.active'))
369     var activeIndex = this.getItemIndex(this.$active = this.$element.find('.item.active'))
370     if (pos > (this.$items.length - 1) || pos < 0) return
371     if (this.isSliding) return this.$element.one('slid.bs.carousel', function () { that.to(pos) })
372     if (this.isSliding) return this.$element.one('slid.bs.carousel', function () { that.to(pos) })
373     if (activeIndex == pos) return this.pause().cycle()
374     if (activeIndex > pos) return this.pause().cycle()
375     if (activeIndex < pos) return this.pause().cycle()
```

JS TS

JavaScript

- JavaScript is the Programming Language for the Web
- Can be used on server-side
 - `Node.js`
- [Image Source](#)



TypeScript

- TypeScript is a syntactic superset of JavaScript which adds static typing.
- It shares the same base syntax as JavaScript, but adds something to it.
- *Much beter than JavaScript, trust me.*

What is a variable?

- A variable is a *container* for a value
- Things we can store.
 - Actual value: `1` / `"Text"` / `true` , ...
 - `Reference` (pointer) to an object

Declaring a variable

```
let myName;  
let myAge;
```

- All code instructions should end with a semi-colon.

Initializing a variable

```
let myName;  
myName = "Chris";
```

or just

```
let myName = "Chris";
```

Note about `var`

```
var myName;  
var myAge;
```

- Old way of declaring variable
- Error prone
 - **Hoisting**
 - Allowing re-declarations
- *Not recommended*

Re-declaration

- (Use quokka)

```
let x = 5;  
let x = 10;  
console.log(x); \\SyntaxError: Identifier 'x' has already been declared
```

Updating a variable

```
let myName = "Chris";  
myName = "Bob"; //Updating
```

const

- Variables defined with const cannot be redeclared and updated.

```
const a = 10;  
// Cannot change a. This will give error.  
// a = 20
```

Good variable naming

Good

```
age;  
myAge;  
init;  
initialColor;  
finalOutputValue;  
audio1;  
audio2;
```

Bad variable naming

```
1;  
a;  
_12;  
myage;  
MYAGE;  
var Document;  
skjfndskjfnbdskjfb;  
thisisareallylongstupidvariablenameman;
```

Variable types

- Numbers
- Strings
- Booleans
- Arrays
- Objects

Numbers

```
let myAge = 17;  
console.log(typeof myAge); //number
```

Number operations

- `+`, `-`, `*`, `/`
- Modulo `%`
- Exponent `**`

Operator precedence

```
const num1 = 10;  
const num2 = 50;  
num2 + num1 / 8 + 2; //Get 53.25  
(num2 + num1) / (8 + 2); //Get 6
```

Increment and decrement operators

```
let a = 1;  
a++;  
console.log(a); // 2  
++a;  
console.log(a); // 3  
a += 1;  
console.log(a); // 4
```

Strings

- Strings are pieces of text.
- When you give a variable a string value, you need to wrap it in single or double quote marks.

```
let dolphinGoodbye = "So long and thanks for all the fish";  
typeof dolphinGoodbye; //string
```

Use quotes in string

```
const bigmouth = "Using ' is okay. Also is \"...\"";  
console.log(bigmouth);  
// Using ' is okay. Also is "..."
```

Concatenating strings

- `+` operator

```
const str1 = "Hello";  
const str2 = "World";  
console.log(str1 + " " + str2 + " !");
```

- Template literal

```
console.log(`${str1} ${str2} !`);
```

String / Number transformation

- A problem with a number stored as a string type

```
let myNumber = "74";  
myNumber += 3; // Get 743
```

- To fix this

```
Number(myNumber) + 3;
```

Multiline strings

- Break characters `\n`

```
const output = "I like the song.\nI gave it a score of 90%.";
console.log(output);

// I like the song.
// I gave it a score of 90%.
```

Multiline strings (cont)

- Template literals respect the line breaks in the source code.

```
const output = `I like the song.  
I gave it a score of 90%.`;
console.log(output);  
  
// I like the song.  
// I gave it a score of 90%.
```


Strings as objects

- Most things are objects in JavaScript.

```
const string = "This is my string";
```

- The variable becomes a `String` object instance.
 - Contains [properties and methods](#).

String length

```
const browserType = "mozilla";  
browserType.length; // 7
```

Accessing string characters

```
browserType[0]; // "m"  
browserType[browserType.length - 1]; // "a"
```

- Remember: computers count from 0, not 1!

Testing if a string contains a substring

```
browserType.includes("zilla"); // true
```

Extracting a substring from a string

```
browserType.slice(1, 4); // "ozi"  
browserType.slice(2); // "zilla"  
browserType.slice(0, -1); // "mozill
```

Changing case

```
const radData = "My NaMe Is MuD";  
console.log(radData.toLowerCase()); //my name is mud  
console.log(radData.toUpperCase()); //MY NAME IS MUD
```

Updating parts of a string

```
const browserType = "mozilla";  
const updated = browserType.replace("moz", "van");  
  
console.log(updated); // "vanilla"  
console.log(browserType); // "mozilla"
```

- Note that `replace()` doesn't change the string it was called on.

Booleans

- Booleans are `true` / `false` values.
- These are generally used to test a condition, after which code is run as appropriate.

```
let iAmAlive = true;  
let test = 6 < 3;  
typeof test; //boolean
```


`null` and `undefined`

- `undefined`
 - Variable has been declared but has not yet been assigned a value.
- `null`
 - Assignment of no value.

```
let testUndefined;  
console.log(testUndefined); //shows undefined  
console.log(typeof testUndefined); //shows undefined
```

```
let testNull = null;  
console.log(testNull); //shows null  
console.log(typeof testNull); //shows object
```