ECMAScript

'

much easy

WoW

ECMAScript 6 Quiz

by @mxwllt

future

v

ction () {}

simple
inside

r u a dev?

retro

much hype

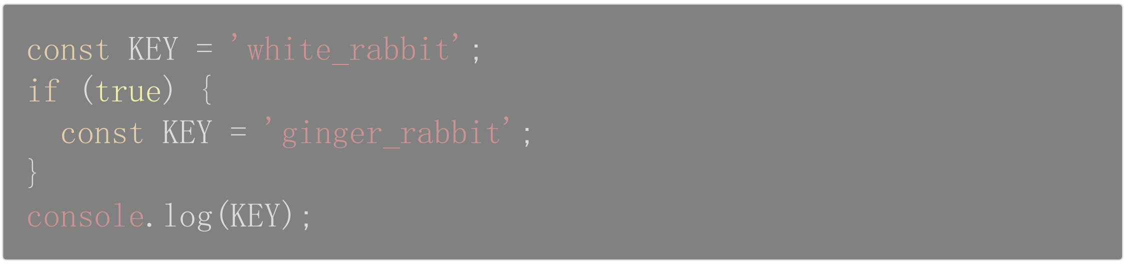
so fab'

compatible
4 evr

WARNING
This quiz is subject to dirty tricks.
Do not trust the speaker.
But don't be scared to answer.

var/let/const

? *What's the output?*



```
const KEY = 'white_rabbit';
if (true) {
  const KEY = 'ginger_rabbit';
}
console.log(KEY);
```

white_rabbit

? *What's the output?*

```
let x = 42;
if (true) {
  let x = 1337;
}
console.log(x);
```

42

? *What's the output?*

```
let x = 42;
if (true) {
  console.log(x);
  let x = 1337;
}
```

ERROR!!!!!!

Functions

```
// Here is a new way to define functions
var double = i => i*2;

// Which is like
var double = function (i) {
  return i * 2;
};
```

? *How to use the new syntax to define a function which doesn't take any* *parameter?*

() => this.whatever;

Templates

? *What's the output?*

```
var x = `foo ${y}`,
    y = `bar ${x}`;

console.log(x);
```

foo undefined

? *What's the output?*

```
var x = `foo ${y}`,
    y = `bar ${x}`;

console.log(y);
```

bar foo undefined

Map and Set

Map example

```
var myMap = new Map(),
    keyObj = {},
    keyFunc = function () {};

myMap.set(keyObj, "value for keyObj");
myMap.set(keyFunc, "value for keyFunc");

myMap.get(keyObj);  // "value for keyObj"
myMap.get(keyFunc); // "value for keyFunc"
```

Set example

```
var mySet = new Set();

mySet.add(5);
mySet.add("something");

mySet.has(5);                // true
mySet.has("some" + "thing"); // true
mySet.has(32);               // false
```

? *What's the attribute to get the quantity of objects stored in a `Set` object?*

length size weight area

size

? *What's the difference between Map/Set and WeakMap/WeakSet ?*

Values cannot be primitive types. Values are objects and are "weakly" held

Declare

? *In ES6, is there a better way to create this object?*

```
let options = {
  protocol: protocol,
  url: url,
  method: method,
  callback: callback
};
```

```
let options = {
  protocol,
  url,
  method,
  callback
};
```

Class

? *Which keywork is not allowed in ES6 Class definition?*

private static constructor set

Private

Class example

```
// Create a logger facade
class Logger {
  constructor (type = "Info") {
    this.type = type;
  }

  get current()     { return `Logger: ${this.type}`; }
  set current(type) { this.type = type; }

  static create(type) { return new this(type); }
  log (message)       { /* Basic method */ }
}
```

Is it legal or not?

Syntax test

```
var score = [12, 7, 14];
Math.max(...score);
```

Yes

```
function stuff(x::Number, y::String) {
  // Do stuff..
}
```

Nope.

```
function stuff(x, ...y) {
  // Do stuff..
}
```

Yes

```
function stuff(x,  y=12) {
  // Do stuff..
}
stuff(2);
```

Yes

```
function stuff(x, y=x/3) {
  // Do stuff..
}
stuff(6);
```

Yes

```
var {foo, bar} = {
  foo: ’FOO’,
  bar: ’BAR’
};
```

Yes

```
var [first, , last] = [1,2,3];
```

Yes

```
function stuff(a, x=12, y=42) {
  // Do stuff..
}
```

```
}
stuff(1,  , 2);
```

Nope.

Promises

? *What the Promise constructor take as parameter(s)?*

Only one parameter: an 'executor' function (resolve, reject) {...}

? *How to use the progress on a ES6 Promise?*

(like in the Q library)

Sorry,
this is not available in ES6 :-(

Progress example in Q

```
uploadFile()
  .then(function (data)     {/* Success */},
        function (err)      {/* Error */},
        function (progress) {/* Progress */});
// Or..
uploadFile()
  .progress(function (progress) {
    // Progress
  });
```

? *Is there a method to take a lot of promises, wrap them into one which will be resolved (or rejected) once one of them will be resolved?*

Yes, it's 'race'

Promise.race()

```
Promise
  .race([loadSiteMap, loadGOTin4K])
  .then(function(value) {
    // Success case
  }, function(reason) {
    // Error case
  });
```

General

General

? *Which one of this keywords won't be compatible on ES6?*

typeof continue for..in

NONE! ES6 is completely retrocompatible

? *Which stable browser got the best ES6 support ?*

Firefox 36 (64%)

Final boss

? *Which var got the max value?*

```
var a = 0x1101010110111,
    b = 0o1101010110111,
    c = 0b1101010110111,
    d = 001101010110111,
    e = 0E1101010110111,
    f =   1101010110111;
```

A

x	Hexa
o/O	Octal
b	Binary
E	x10exp

More info at

MDN: Mozilla Developer Network

https://developer.mozilla.org/

ES6 compatibility table (by Kangax)

http://kangax.github.io/compat-table/es6/

Power of ES6 (by Charles B. King)

http://charlesbking.com/power_of_es6/#/45

Thanks everybody