



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

React Native cơ bản

Bài 2: Lập trình JavaScript (ES6)

<http://csc.edu.vn/laptrinh/>

Lập trình JavaScript (ES6)

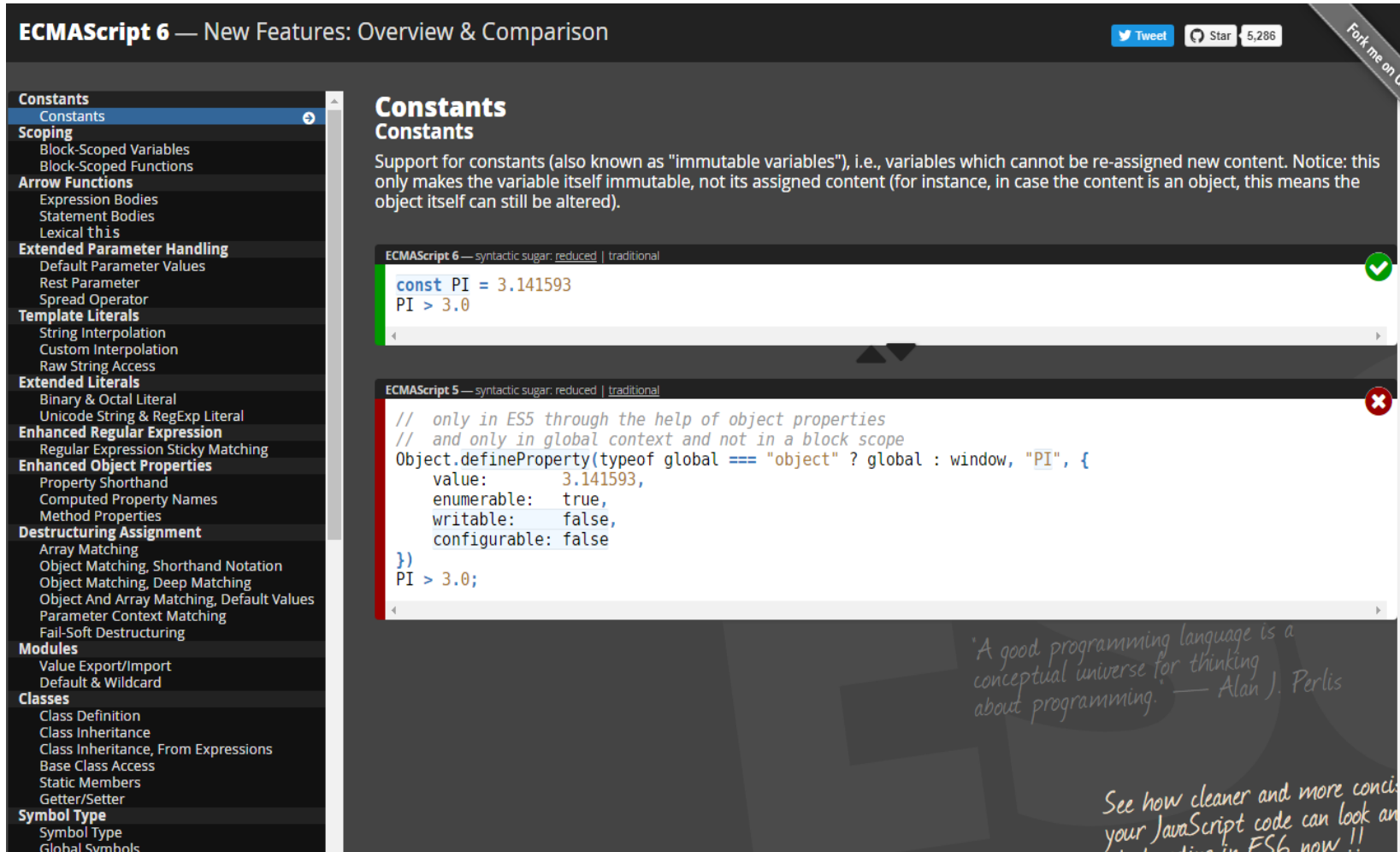


- ☐ Giới thiệu
- ☐ Ngôn ngữ JavaScript
- ☐ Xử lý bất đồng bộ
- ☐ Minh họa

- ES 6 (ECMAScript 6) là phiên bản mới của JavaScript được tích hợp thêm các tính năng và kỹ thuật nâng cao.
- Được ECMAScript phát hành vào năm 2015
- Hiện nay được lập trình trong các loại ứng dụng Web, Desktop, Mobile



Tài liệu: <http://es6-features.org>



The screenshot shows the 'ECMAScript 6 — New Features: Overview & Comparison' page. On the left is a sidebar with a tree view of features: Constants, Scoping, Arrow Functions, Extended Parameter Handling, Template Literals, Extended Literals, Enhanced Regular Expression, Enhanced Object Properties, Destructuring Assignment, Modules, and Classes. The 'Constants' section is selected. The main content area is titled 'Constants' and explains that constants (also known as 'immutable variables') are variables that cannot be re-assigned new content. It notes that this only makes the variable itself immutable, not its assigned content. Below the text are two code snippets. The first, labeled 'ECMAScript 6', shows the modern syntax: `const PI = 3.141593` and `PI > 3.0`, marked with a green checkmark. The second, labeled 'ECMAScript 5', shows the older, more verbose syntax using `Object.defineProperty` to create a read-only property on the global object, marked with a red X. At the bottom right, there is a quote by Alan J. Perlis: 'A good programming language is a conceptual universe for thinking about programming.' and a note: 'See how cleaner and more concise your JavaScript code can look and be in ES6 now!!'.

ECMAScript 6 — New Features: Overview & Comparison

Constants

Support for constants (also known as "immutable variables"), i.e., variables which cannot be re-assigned new content. Notice: this only makes the variable itself immutable, not its assigned content (for instance, in case the content is an object, this means the object itself can still be altered).

ECMAScript 6 — syntactic sugar: **reduced** | traditional

```
const PI = 3.141593
PI > 3.0
```

ECMAScript 5 — syntactic sugar: **reduced** | **traditional**

```
// only in ES5 through the help of object properties
// and only in global context and not in a block scope
Object.defineProperty(typeof global === "object" ? global : window, "PI", {
  value:      3.141593,
  enumerable: true,
  writable:   false,
  configurable: false
})
PI > 3.0;
```

"A good programming language is a conceptual universe for thinking about programming." — Alan J. Perlis

See how cleaner and more concise your JavaScript code can look and be in ES6 now!!

❑ Ưu điểm

- Dễ học
- Phát triển ứng dụng loại Web, Desktop, Mobile
- Xây dựng Dịch vụ trên các môi trường Window, Linux

❑ Khuyết điểm

- Chưa được hỗ trợ tốt trên một số trình duyệt
- Thư viện hàm cơ sở chưa đầy đủ và thống nhất

Ngôn ngữ JavaScript



- ☐ Biến và hằng
- ☐ Cấu trúc điều khiển
- ☐ Hàm

❑ Biến

- Cú pháp: [Từ khóa] <Tên biến>

- let: khai báo cục bộ
- var: khai báo toàn cục

- Ví dụ:

```
<script>  
    let ho_ten='Nguyễn Văn A';  
    console.log('Họ tên :' + ho_ten);  
    var tuoi=25;  
    console.log('Tuổi của bạn :' + tuoi);  
</script>
```

- Kết quả:

Họ tên: Nguyễn Văn A

Tuổi của bạn: 25

❑ Biến danh sách (Array)

- Cú pháp:

[Từ khóa] <Tên biến> =[]

hoặc [Từ khóa] <Tên biến> =[giá trị 1, giá trị 2,...]

- Ví dụ:

```
var Danh_sach_A=[];  
console.log('Số phần tử trong Danh sách A: ' + Danh_sach_A.length);  
var Danh_sach_B=[12,3,5,7,8,19];  
console.log('Số phần tử trong Danh sách B: ' + Danh_sach_B.length);  
// Duyệt Danh sách  
Danh_sach_B.forEach(n=>{  
|   console.log(n)  
})
```


❑ Biến danh sách (Array)

- Kết quả

Số phần tử trong Danh sách A: 0

Số phần tử trong Danh sách B: 6

12

3

5

7

8

19

❑ Biến danh sách (Array)

- Phương thức sort: sắp xếp các phần tử trong danh sách
- Ví dụ:

```
Danh_sach_Nhom_Tivi=[
    {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
    {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
    {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
    {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
// Sắp xếp số
Danh_sach_Nhom_Tivi.sort((a,b)=>{
    return Number(b.Don_gia)-Number(a.Don_gia);
});
Danh_sach_Nhom_Tivi.forEach(tv => {
    console.log(`${tv.Ma_so} - ${tv.Don_gia}`)
});
```

- Kết quả:

```
LG - 500000
SONY - 450000
KHAC - 250000
SAMSUNG - 150000
```

❑ Biến danh sách (Array)

- Phương thức sort: sắp xếp các phần tử trong danh sách
- Ví dụ:

```
Danh_sach_Nhom_Tivi=[
    {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
    {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
    {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
    {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
// Sắp xếp Chuỗi
Danh_sach_Nhom_Tivi.sort((a, b) => a.Ma_so.localeCompare(b.Ma_so))
Danh_sach_Nhom_Tivi.forEach(tv => {
    console.log(`${tv.Ma_so} - ${tv.Don_gia}`)
});
```

- Kết quả:

```
KHAC - 250000
LG - 500000
SAMSUNG - 150000
SONY - 450000
```

❑ Biến danh sách (Array)

- Phương thức push: thêm 1 phần tử vào trong danh sách
- Ví dụ:

```
Danh_sach_Nhom_Tivi=[
  {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
  {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
  {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
  {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
let Tivi={
  "Ma_so":"TOSHIBA","Ten":"Tivi TOSHIBA","Don_gia":150000
}
Danh_sach_Nhom_Tivi.push(Tivi)
Danh_sach_Nhom_Tivi.forEach(tv => {
  console.log(`${tv.Ma_so} - ${tv.Don_gia}`)
});
```

- Kết quả:

```
LG - 500000
SAMSUNG - 150000
SONY - 450000
KHAC - 250000
TOSHIBA - 1500000
```

❑ Biến danh sách (Array)

- Phương thức find: tìm 1 phần tử trong danh sách
- Ví dụ:

```
Danh_sach_Nhom_Tivi=[
  {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
  {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
  {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
  {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
var gtTim='lg';
let kqTim=Danh_sach_Nhom_Tivi.find(x=>x.Ma_so.toLowerCase()==gtTim.toLowerCase())
console.log(kqTim);
```

- Kết quả:

```
Object {Ma_so: "LG", Ten: "Tivi LG", Don_gia: 500000}
```

- Lưu ý: nếu tìm không thấy gtTim là **undefined**

❑ Biến danh sách (Array)

- Phương thức findIndex: tìm 1 phần tử trong danh sách
- Ví dụ:

```
Danh_sach_Nhom_Tivi=[  
  {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},  
  {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},  
  {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},  
  {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}  
];  
var gtTim='samsung';  
let chiso=Danh_sach_Nhom_Tivi.findIndex(x=>x.Ma_so.toLowerCase()==gtTim.toLowerCase())  
console.log(chiso);
```

- Kết quả:

1

- Lưu ý: nếu tìm không thấy chỉ số là -1

❑ Biến danh sách (Array)

- Phương thức filter, map, reduce:
- Ví dụ filter:

```
Danh_sach_Nhom_Tivi=[
  {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
  {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
  {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
  {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
var gtTim='s'
dsLoc=Danh_sach_Nhom_Tivi.filter(x => x.Ten.toLowerCase().includes(gtTim.toLowerCase()))
if(dsLoc.length>0){
  console.log(dsLoc)
}else{
  console.log('Không tìm thấy')
}
```

- Kết quả trả về là một mảng:

```
> Array(2) [Object, Object]
```

- Lưu ý: nếu tìm không thấy trả về một mảng có chiều dài là 0

❑ Biến danh sách (Array)

- Phương thức filter, map, reduce:

- Ví dụ map:

```
Danh_sach_Nhom_Tivi=[
  {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
  {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
  {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
  {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
// Giảm 10% Đơn giá cho tất cả các tivi
dsKhuyenmai=Danh_sach_Nhom_Tivi.map((Tivi)=>{
  Tivi.Don_gia*=0.9
  return Tivi
})
dsKhuyenmai.forEach(tv => {
  console.log(`${tv.Ma_so} - ${tv.Don_gia}`)
})
```

- Kết quả trả về là một mảng có số phần tử bằng với số phần tử mảng gốc:

```
LG - 450000
SAMSUNG - 135000
SONY - 405000
KHAC - 225000
```


❑ Biến danh sách (Array)

- Phương thức filter, map, reduce:
- Ví dụ reduce:

```
Danh_sach_Nhom_Tivi=[
  {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
  {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
  {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
  {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
// Tính tổng Đơn giá cho tất cả các ti vi
let tongDongia=Danh_sach_Nhom_Tivi.reduce((tong,Tivi)=>{
  tong+=Tivi.Don_gia
  return tong
},0)
console.log(`Tổng Đơn giá: ${tongDongia}`)
```

- Kết quả trả về là một giá trị:

Tổng Đơn giá: 1350000

❑ Biến danh sách (Array)

- Phương thức splice: xóa 1 phần tử trong danh sách
- Ví dụ:

```
Danh_sach_Nhom_Tivi=[
  {"Ma_so":"LG","Ten":"Tivi LG","Don_gia":500000},
  {"Ma_so":"SAMSUNG","Ten":"Tivi Samsung","Don_gia":150000},
  {"Ma_so":"SONY","Ten":"Tivi Sony","Don_gia":450000},
  {"Ma_so":"KHAC","Ten":"Tivi Khác","Don_gia":250000}
];
var gtTim='samsung';
let chiso=Danh_sach_Nhom_Tivi.findIndex(x=>x.Ma_so.toLowerCase()==gtTim.toLowerCase())
if(chiso!=-1){
  Danh_sach_Nhom_Tivi.splice(chiso,1);
}
```

- Kết quả:

```
LG - 500000
SONY - 450000
KHAC - 250000
```

❑ Hằng

- Cú pháp: `const <Tên hằng>=<Trị>`
- Ví dụ:

```
const PI=3.1416  
console.log(`Hằng số PI:${PI}`)
```

- Kết quả:

```
Hằng số PI:3.1416
```

❑ Cấu trúc if

- Cú pháp 1:

```
if (điều kiện){  
    // Lệnh xử lý  
}
```

- Cú pháp 2:

```
if (điều kiện){  
    // Lệnh xử lý 1  
}  
else{  
    // Lệnh xử lý 2  
}
```

❑ Cấu trúc if

- Ví dụ:

```
let diemTb=6.5
let xetKq=''
if(diemTb>=5){
  xetKq='Đậu'
}else{
  xetKq='Rớt'
}
console.log(`Kết quả cuối năm: ${xetKq}`)
```

- Kết quả:

Kết quả cuối năm: Đậu

❑ Cấu trúc switch

- Cú pháp :

```
switch (biểu thức){  
  case giá trị 1:  
    // Lệnh xử lý 1  
    break;  
  case giá trị 1:  
    // Lệnh xử lý 2  
    break;  
  default:  
    // Lệnh xử lý n  
}
```

❑ Cấu trúc switch

- Ví dụ:

```
let masoTivi = `LG`
let tenTivi = ``
switch (masoTivi) {
  case "LG":
    tenTivi = `Tivi LG`
    break
  case "SAMSUNG":
    tenTivi = `Tivi Samsung`
    break
  case "SONY":
    tenTivi = `Tivi Sony`
    break
  case "KHAC":
    tenTivi = `Tivi Khác`
    break
}
console.log(`Tên: ${tenTivi}`)
```

- Kết quả:

Tên: Tivi LG

❑ Xây dựng hàm

- Cú pháp :

```
[Từ khóa] <Tên hàm> = ([Danh sách tham số]) => {  
    // Lệnh xử lý  
}
```

- Ví dụ: xây dựng hàm tính tổng trị các phần tử trong danh sách

```
let Tinh_tong=(Danh_sach)=>{  
    let kq=0  
    Danh_sach.forEach(x=>{  
        kq+=x;  
    })  
    return kq;  
}
```


❑ Gọi thực hiện

- Cú pháp :

<Tên hàm> ([Danh sách trị])

- Ví dụ: gọi thực hiện hàm Tính tổng

```
let dsSo=[1,2,3,4,5]
// Gọi Hàm
let kq=Tinh_tong(dsSo)
console.log(`Tổng [ ${dsSo.join()}] là: ${kq}`)
```

- Kết quả:

Tổng [1,2,3,4,5] là: 15

❑ Promise

- Xử lý các tác vụ bất đồng bộ, sử dụng callback function khi kết quả thành công hoặc thất bại

- Khai báo :

```
function Hình_chu_nhat(d,r){  
    return new Promise((kq,loi)=>{  
        if(isNaN(d) || isNaN(r)){  
            loi('Dữ liệu không hợp lệ');  
        }else{  
            let dt=d*r;  
            let cv=(d+r)*2  
            let hcn={  
                dientich:dt,  
                chuvi:cv,  
            }  
            kq(hcn);  
        }  
    })  
}
```

- Gọi thực hiện:

```
Hình_chu_nhat(20,5).then(hcn=>{  
    console.log(hcn.dientich)  
    console.log(hcn.chuvi)  
}).catch(loi=>{  
    console.log(loi)  
})
```

Xử lý bất đồng bộ



- Kết quả:

Diện tích: 100

Chu vi: 50

❑ async

- async function định nghĩa một hàm bất đồng bộ, kết quả trả về một đối tượng async function
- Khai báo :

```
// Xây dựng Promise
function Tim_so_lon(a,b){
    return new Promise((kq,loi)=>{
        let so_lon=a>b?a:b;
        kq(so_lon)
    })
}

// Xây dựng async - await
async function Tim_so_lon_ba_so(a,b,c){
    let kq1= await Tim_so_lon(a,b);
    console.log(kq1)
    let kq2= await Tim_so_lon(kq1,c);
    console.log(kq1)
    return kq2;
}
```

Xử lý bất đồng bộ



- Gọi thực hiện

```
// Gọi
Tim_so_lon_ba_so(10,45,23).then(kq=>{
  console.log(`Số lớn: ${kq}`)
})
```

- Kết quả:

```
45
45
Số lớn: 45
```

- Giáo viên minh họa