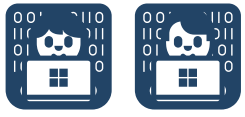


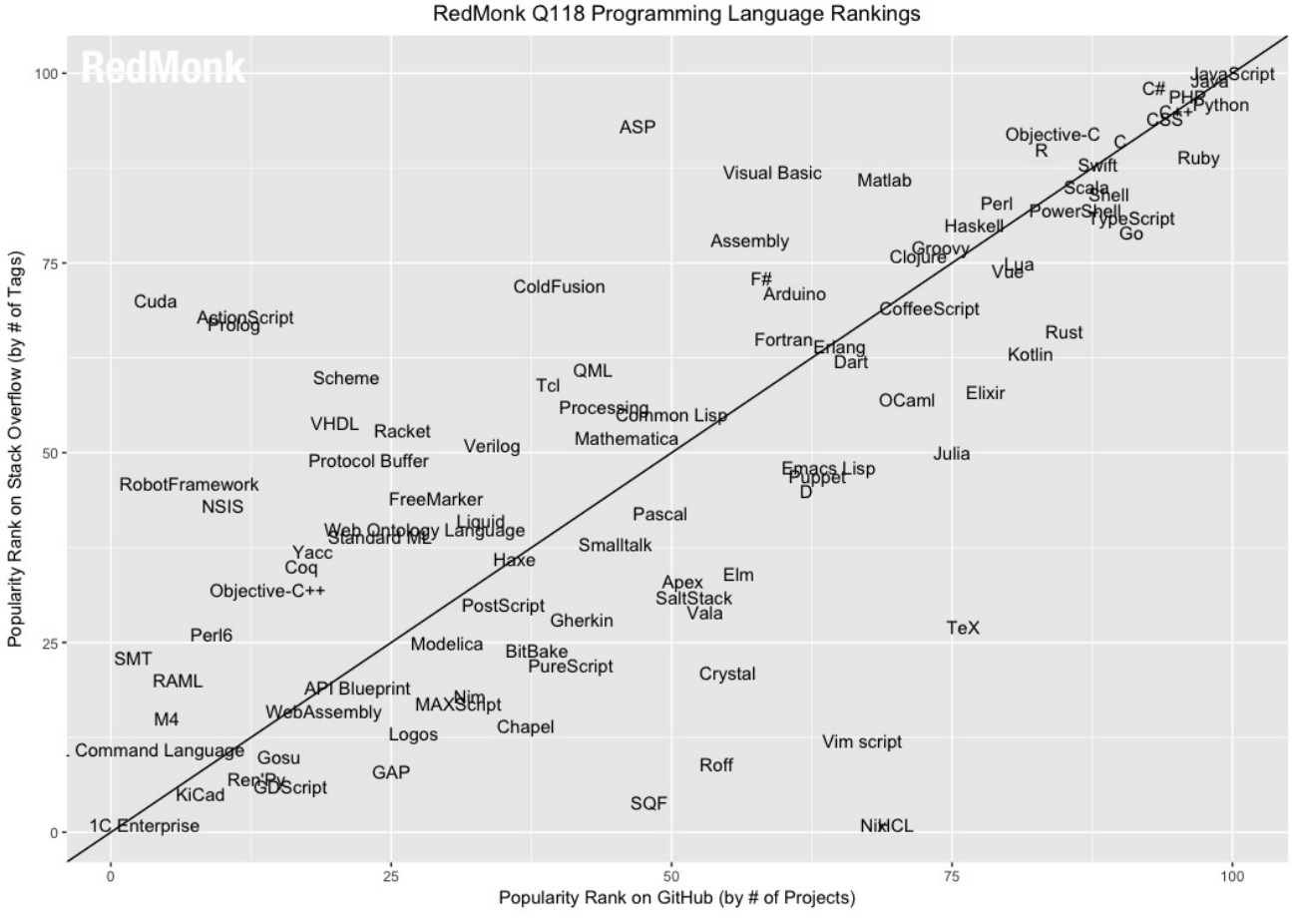
DoiT Web Project Team

6. JavaScript: First Steps

JavaScript

- 자바(Java)와는 관련이 없다.
- 프론트엔드에서 동적인 웹 콘텐츠를 구현하기 위한 언어.
- ...였으나 백엔드, PC, 모바일 등 훨씬 많은 곳에 쓰이게 됨.
- 너무 자유롭고 유연한 언어.

JavaScript



꼭 알아야해요!

JavaScript의 특성

- **인터프리트 언어:** 인터프리터가 코드를 읽는 즉시 실행.
- **순차적인 실행:** 위에서 아래로 순차적으로 코드를 실행.
- **클라이언트 사이드 언어:** 사용자 단말에서 코드를 실행.

ECMAScript

- ECMA International에서 자바스크립트 표준화 작업을 함.
- ECMAScript는 ECMA 명세에 따라 구현한 표준 스크립트 언어.
- ECMAScript가 자바스크립트의 토대를 구성한다.
- 대부분의 브라우저가 ES6(ECMAScript 2015)까지 지원.

🔗 HTML 파일에 연결하기

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    ...
    <script type="text/javascript">
      console.log('Hello, world!');
    </script>
  </body>
</html>
```

- `script` 태그 안에 자바스크립트 코드를 넣는 방식.
- 별로 추천하지 않는다.

HTML 파일에 연결하기

```html
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    ...
    <script src="index.js"></script>
  </body>
</html>
```

```js
// index.js
console.log('Hello, world!');
```

- 별도의 외부 자바스크립트 파일을 불러오는 방식.
- 브라우저에서 html 파일을 열고 F12 - Console을 확인해보자.

📖 Dynamic Typed

```
// function scope variable
var foo = 10;
```

```
// block scope variable
let foo = 10;
```

```
// block scope constant
const foo = 10;
```

- `let` , `const` 는 ES6 문법, 해석하지 못하는 브라우저가 있음.
- 크로스 브라우징을 위해서는 transpiler를 사용해야 한다.

📖 Primitive Data Type

```
// number
var foo = 10;
```

```
// string
var foo = 'hello';
```

```
// boolean
var foo = true;
```

📖 Primitive Data Type

```
// undefined
var foo = undefined;
```

```
// null
var foo = null;
```

```
// symbol (ES6)
var foo = Symbol('description');
```

📖 Operators

- 덧셈: `a + b`
- 뺄셈: `a - b`
- 곱셈: `a * b`
- 나눗셈: `a / b`
- 나머지: `a % b`
- 제곱: `a ** b` (ES7)
- 증가: `a++`
- 감소: `a--`

📖 Operators

- 덧셈 복합대입: `a += b`
- 뺄셈 복합대입: `a -= b`
- 곱셈 복합대입: `a *= b`
- 나눗셈 복합대입: `a /= b`
- 나머지 복합대입: `a %= b`
- 제곱 복합대입: `a **= b` (ES7)

📖 Operators

- Grater than: `a > b`
- Greater than or equal: `a >= b`
- Less than: `a < b`
- Less than or equal: `a <= b`
- 논리 AND: `a && b`
- 논리 OR: `a || b`
- 논리 NOT: `!a`
- Bitwise: `<<`, `>>`, `|`, `&`

📖 Operators

- 느슨한 같음 비교: `a == b`
- 엄격한 같음 비교: `a === b`
- 느슨한 다름 비교: `a != b`
- 엄격한 다름 비교: `a !== b`

```
var a = '10';
var b = 10;

console.log(a == b); // true
console.log(a === b); // false
```

📖 Conditional Statement

```
// if ~ else
if (a === b) { ... } else { ... }
```

```
// if ~ else if
if (a === b) { ... } else if (a === c) { ... }
```

```
// Ternary
a === b ? (...) : (...)
```

📖 Loop

```
// while
while (condition) { ... }
```

```
// do ~ while
do { ... } while (condition);
```

```
// for
for (var i = 0; i < 10; i += 1) { ... }
```

📖 Object

```
var obj = {
  foo: 'value',
  bar: 'value',
  ...
};
```

- 프로퍼티(Property)의 집합.
- 각 프로퍼티는 key-value pair (KVP)
- 소프트웨어적으로 현실 세계를 표현하기 적합하다.

📖 Object

Dot Notation

```
var cat = {
  name: 'Cake',
  age: 5,
};

console.log(cat.name); // 'Cake'
console.log(cat.age); // 5
```

- 마침표를 이용해 프로퍼티에 접근할 수 있다.

📖 Object

Bracket Notation

```
var key = 'name';
var cat = {
  name: 'Cake',
  age: 5,
};

console.log(cat[key]); // 'Cake'
key = 'age';
console.log(cat[key]); // 5
```

- 대괄호를 이용해 동적으로 프로퍼티에 접근할 수 있다.

📖 Object

Destructuring

```
var cat = {
  name: 'Cake',
  age: 5,
};

var { name } = cat; // var name = cat.name;
console.log(name); // 'Cake'
```

- Destructuring 구문으로 프로퍼티 값을 할당할 수 있음.
- ES6 문법이므로 아직 해석하지 못하는 브라우저가 있다.

📖 Array

```
var arr = [1, 2, 3];
```

- 사실 자바스크립트에서 배열은 객체.
- `push`, `pop`, `length`와 같은 프로퍼티를 가지고 있다.

📖 Function

```
function doSomething() {
  ...
}
```

- 함수 선언문
- 함수 표현식
  - 익명 함수 표현식
  - 기명 함수 표현식
  - 즉시 실행 함수 표현식 (IIFE)

📖 Function

함수 선언문

```
function square(num) {
  return num * num;
}

console.log(square(2)); // 4
```

- 이름이 붙여진 함수를 기명 함수라고 한다.
- 다른 언어의 일반적인 함수 사용법과 동일.

📖 Function

함수 표현식

익명 함수 표현식

```
var square = function (num) {
  return num * num;
};

console.log(square(2)); // 4
```

- 이름이 없는 함수를 익명 함수라고 한다.
- 변수에 함수를 할당할 수 있다.

📖 Function

함수 표현식

기명 함수 표현식

```
var math = {
  factorial: function f(n) {
    if (n <= 1) {
      return 1;
    }

    return n * f(n - 1);
  }
};

console.log(math.factorial(3)); // 6
```

- 해당 함수의 body 안에서만 사용할 수 있다.

📖 Function

함수 표현식

즉시 실행 함수 표현식

```
(function () {
  console.log('Cake the Cat');
})();
```

- 로드된 즉시 호출, 실행되는 함수.

📖 Function

함수 표현식

즉시 실행 함수 표현식

```
(function (name, species) {
  console.log(name + ' the ' + species);
})('Cake', 'Cat');
```

- 파라미터를 전달할 수도 있다.

📖 Function

- 자바스크립트에서 함수는 **일급 객체** (First-class citizen)
  - 데이터 구조에 담을 수 있음.
  - 파라미터로 전달할 수 있음.
  - 반환 값으로 사용할 수 있음.

📖 Function

데이터 구조에 담기

```
var square = function(num) {
  return num * num;
};

console.log(square(2)); // 4
```

- 변수에 함수를 할당할 수 있다.

📖 Function

데이터 구조에 담기

```
var cat = {
  name: 'Cake',
  age: 5,
  makeNoise: function() { // method
    console.log('Meow!');
  },
};

cat.makeNoise(); // 'Meow!'
```

- 프로퍼티에 할당된 `makeNoise`를 **메소드**(Method)라고 부른다.

📖 Function

파라미터로 전달하기

```
function doSomething(callback) {
  callback();
}

function makeNoise() {
  console.log('Meow!');
}

doSomething(makeNoise); // 'Meow!'
```

- 인자로 전달된 `makeNoise`를 **콜백**(Callback)이라고 한다.

📖 Function

반환 값으로 사용하기

```
var makeNoise = function (name, noise) {
  var text = name + ': ' + noise;
  return function () {
    return text;
  }
};

var cakeNoise = makeNoise('Cake', 'Meow!');
console.log(cakeNoise()); // 'Cake: Meow!'
```

- 스코프를 기억하는 내부함수가 **클로저**(Closure)를 형성했다.

📖 JavaScript OOP

```
function Cat(name, age) { // constructor function (class)
  this.name = name; // member property
  this.age = age;

  this.makeNoise = function () { // method
    console.log('Meow!');
  }
}

var cake = new Cat('Cake', 5);
cake.makeNoise(); // 'Meow!'
```

- 조금 특이한 방식을 사용해야 한다.
- 다른 OOP언어에서 쓰는 `class` 문법은 ES6에 등장.

📖 JavaScript OOP

`this` 키워드

```
function Cat(name, age) {
  this.name = name;
  this.age = age;

  this.makeNoise = function () {
    console.log('Meow!');
  }
}
```

- `this`는 객체의 멤버를 참조할 때 사용.
- 전역범위에서는 전역객체(브라우저에선 `window`)를 가리킨다.

📖 JavaScript OOP

`prototype` 프로퍼티

```
function Cat(name, age) {
  this.name = name;
  this.age = age;
}

Cat.prototype.makeNoise = function () {
  console.log('Meow!');
}
```

- 모든 함수가 공통으로 가진 프로퍼티.
- `Cat` 인스턴스를 생성할 때 메모리를 절약할 수 있다.

📖 JavaScript OOP

`class` 문법

```
class Cat {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  makeNoise() {
    console.log('Meow!');
  }
}
```

- ES6에 추가된 문법

🤔 그래서 JavaScript로 뭘 할 수 있나요?

```
var el = document.getElementById('button');

el.addEventListener('click', function() {
  alert('clicked');
});
```

- 동적인 웹 콘텐츠 구현
- 사용자 인터랙션 처리
- 어플리케이션 로직 처리