쇼핑몰을 만드는 프론트엔드 개발

# 자바스크립트 기초

김태곤

# 자바스크립트

- 1995년 5월, 넷스케이프 사에서 브랜든 아이크가 개발했다.
- 언어적으로는 자바와 아무런 상관이 없다.
- 공식적으로 JavaScript라 표기한다.
- 사실 오라클이 상표권을 가진 등록 상표이다. 표준 명세의 이름은 **ECMAScript(또는 ECMA-262)**이다.
- 2015년 6월, **ECMAScript 2015(또는 ECMAScript 6)**의 명세가 확정되었다.
- 현재 가장 널리 사용되는 것은 ECMAScript 5이다.

# 기본 문법

- 대소문자를 구분한다.
- 유니코드를 기반으로 하고 있어서 아스키 코드에 없는 언어로 코드를 작성할 수도 있고 문자열 안에서의 길이도 같다.
- 명령어는 문장(statement)라 부른다.
- 한 문장은 세미콜론(;)으로 끝맺는다.
- 스페이스, 탭, 줄바꿈 문자는 공백(white space)이라 부른다.

# ⟨script⟩

- 브라우저 환경에서 자바스크립트를 사용할 때 필요한 엘리먼트
- CSS처럼 HTML 문서 안에 여닫는 태그 사이에 코드를 입력할 수도 있고 외부의 자바스크립트 파일을 불러올 수도 있다.

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

<script type="text/javascript" src="a.js"></script>
```

- HTML5부터는 type 속성을 생략할 수 있고 생략하면 기본값은 "text/javascript"가 된다.

# ⟨script⟩

- 브라우저 환경에서 자바스크립트를 사용할 때 필요한 엘리먼트
- CSS처럼 HTML 문서 안에 여닫는 태그 사이에 코드를 입력할 수도 있고 외부의 자바스크립트 파일을 불러올 수도 있다.

```
<script>
console.log('Hello, world');
</script>

<script src="a.js"></script>
```

- HTML5부터는 type 속성을 생략할 수 있고 생략하면 기본값은 "text/javascript"가 된다.

# 〈script〉

- 〈script〉는 기본 상태에서는 반드시 순차적으로만 실행된다.
- 브라우저는 〈script〉를 만나면 렌더링을 멈추고 다운로드 및 실행을 끝낸 후 다시 렌더링을 시작한다.

https://stevesouders.com/hpws/move-scripts.php

HTML 해석기

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>자바스크립트 기초</title>
```

자바스크립트 엔진

```
  <script src="path/filename.js"></script>
```

HTML 해석기

```
</head>
<body>
  <p>반갑습니다.</p>
</body>
</html>
```

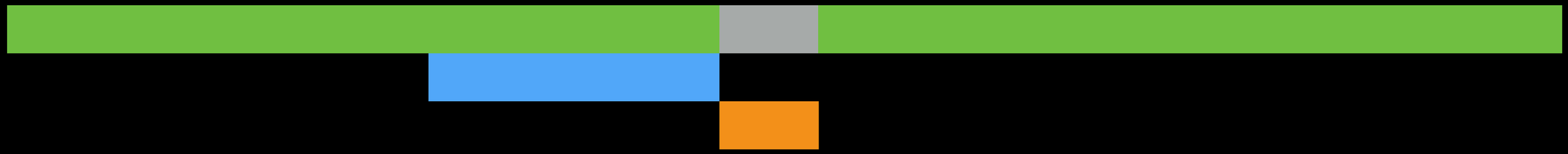
# 〈script〉

- 〈script〉는 기본 상태에서는 반드시 순차적으로만 실행된다.
- 브라우저는 〈script〉를 만나면 렌더링을 멈추고 다운로드 및 실행을 끝낸 후 다시 렌더링을 시작한다.
  https://stevesouders.com/hpws/move-scripts.php
- 따라서, 닫는 〈/body〉 바로 위에 두는 것이 가장 좋다.
- defer나 async 속성을 추가하면 병렬로 읽어들이지만, 실행 순서가 보장되지 않으므로 주의해야 한다.

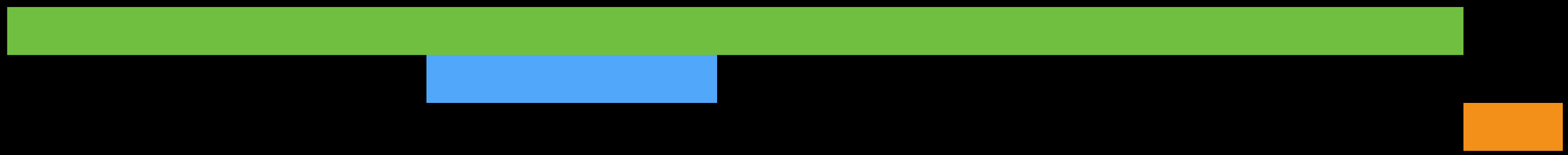
HTML 해석 HTML 해석 정지 스크립트 다운로드 스크립트 실행

〈script〉

〈script async〉



〈script defer〉



http://www.growingwiththeweb.com/2014/02/async-vs-defer-attributes.html

# 〈script〉

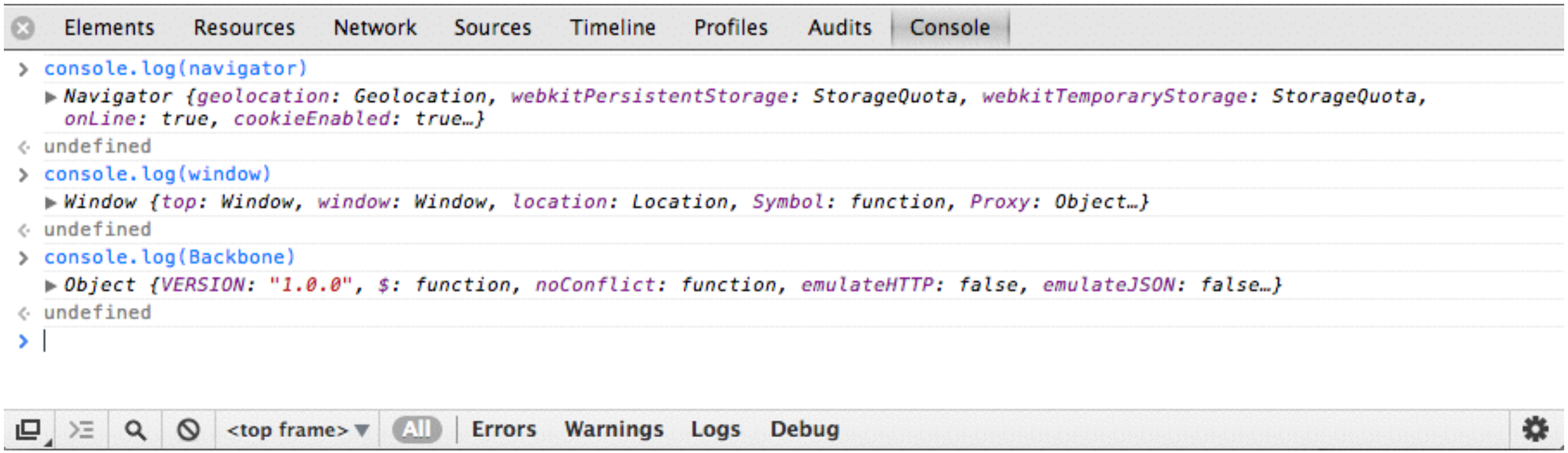
- 〈script〉는 기본 상태에서는 반드시 순차적으로만 실행된다.
- 브라우저는 〈script〉를 만나면 렌더링을 멈추고 다운로드 및 실행을 끝낸 후 다시 렌더링을 시작한다.
  https://stevesouders.com/hpws/move-scripts.php
- 따라서, 닫는 〈/body〉 바로 위에 두는 것이 가장 좋다.
- defer나 async 속성을 추가하면 병렬로 읽어들이지만, 실행 순서가 보장되지 않으므로 주의해야 한다.
- 가능하다면 async를 사용하자.

# Console API

- Chrome, Firefox 등 최신 브라우저에서 지원하는 개발자용 API

  https://developer.chrome.com/devtools/docs/console-api

- 일단은 `console.log(value)` 사용



# 주석(Comment)

- 한 줄 주석 : 슬래시 두 개(//)로 시작. 줄 끝까지 주석으로 인식.
- 블럭 주석 : /* 로 열고 */로 닫음. 여러 줄 주석 가능.

```
// 한 줄 주석입니다.

/*
 여러 줄이 되는 주석입니다.
 주석을 닫는 기호 뭉치만 아니면 무엇이든 입력할 수 있습니다.
 */
```

# 변수(Variables)

- `var + 공백 + 변수이름` 형태로 선언한다.

```
var num;
```

- `변수이름 = 값` 형태로 값을 할당한다.

```
num = 1;
```

- 선언과 할당을 동시에 할 수도 있다.

```
var num = 1;
```

- 쉼표로 구분하면 여러 변수를 선언할 수도 있다.

```
var num = 1, num2 = 2;
```

# 변수(Variables)

- 변수의 값을 다른 변수에 할당할 수 있다.

```
var num = 1, num2 = 2;

num = num2;
console.log(num); // 2
```

- 변수 이름은 $, _를 제외한 공백, 특수 문자, 구두점을 사용할 수 없다. 숫자로 시작할 수 없다. 유니코드 문자도 사용 가능하지만 예약어는 사용할 수 없다.

  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar

- 선언되어 있지 않은 변수 이름에 var 키워드없이 값을 할당하면 전역 변수가 된다.

# 자료형(Data Types)

- 원시 자료형(Primitive Types):
  숫자, 문자열, 불리언, null, undefined
- 객체 자료형(Object Types):
  배열, 오브젝트, 날짜 객체, **함수** 등 원시 자료형을 제외한 전부

In JavaScript, functions are first-class citizens.
자바스크립트에서 함수는 1급 객체이다.

이것만 기억하세요.

= 함수도 다른 자료형처럼 할당, 저장, 복사가 가능하다.

# 원시 자료형

- 해당하는 **리터럴 (literal)** 표현이 있다.

소스 코드에서 고정된 값을 표현할 때 사용.

ex) 1, true, "이름"

# 원시 자료형

- 해당하는 리터럴(literal) 표현이 있다.
- 다른 언어의 원시 자료형과 비슷하지만, 자바스크립트는 모든 값을 객체처럼 다루므로 원시 자료형에도 프로퍼티와 메소드가 존재하는 것처럼 보인다.

```
"Hello, world".length; // 12
```

- 원시 자료형이 저장된 변수를 다른 변수에 할당하면 값 자체가 복사되고 복사된 변수를 변경해도 원래 변수는 변하지 않는다.

```
var color1 = "green";
var color2 = color1;
color2 = "blue";
```

```
var color1 = "green";
```

| Variable Object | |
|---|---|
| color1 | "green" |

```
var color1 = "green";
var color2 = color1;
```

| Variable Object | |
|---|---|
| color1 | "green" |
| color2 | "green" |

```
var color1 = "green";
var color2 = color1;
color2 = "blue";
```

| Variable Object | |
|---|---|
| color1 | "green" |
| color2 | "blue" |

# 원시 자료형

- 해당하는 리터럴(literal) 표현이 있다.
- 다른 언어의 원시 자료형과 비슷하지만, 자바스크립트는 모든 값을 객체처럼 다루므로 원시 자료형에도 프로퍼티와 메소드가 존재하는 것처럼 보인다(null, undefined 제외).

```
"Hello, world".length; // 12
```

- 원시 자료형이 저장된 변수를 다른 변수에 할당하면 값 자체가 복사되고 복사된 변수를 변경해도 원래 변수는 변하지 않는다.

```
var color1 = "green";
var color2 = color1;
color2 = "blue";
```

# 원시 자료형

- typeof 연산자를 사용해서 타입을 확인하기 쉽다.

```
typeof "Hello, world" // "string"
typeof 100 // "number"
typeof undefined // "undefined"
typeof false // "boolean"
typeof null // "object"
```

- 원시 자료형이지만 각 자료형을 표현하는 생성자가 있다.
  String, Number, Boolean

# 숫자(Number)

- 정수, 실수 등을 표현하는 원시 타입
- 리터럴로 표현하는 숫자는 10진수가 기본이나 8진수 16진수 형태도 표현할 수 있다.

```
1234    123.4    0.81    .5    0xFF    0x1f    017
```

- 사칙연산(+, -, *, /), 나머지 연산(%), 증감 연산(++, --), 비트 연산(&, |, ^, ~, >>, <<, >>>), 단항 연산자(+, -)
- 숫자 자료형에는 **NaN** (Not a Number)도 있다.

# 문자열(String)

- 문자열을 표현하는 원시 타입
- 따옴표 또는 작은 따옴표로 묶은 연속된 문자

```
"Hello, world" "문자열" '작은 따옴표'
```

- [ ] 연산자로 특정 인덱스의 글자만 가져올 수 있다.

```
"Hello, world"[1] // "e"
"문자열"[0] // "문"
```

- + 연산자로 문자열 두 개를 연결할 수 있다.

```
"Hello, world" + "는 문자열" // "Hello, world는 문자열"
```

- 사실, 문자열 외의 다른 자료형도 연결할 수 있다.

# 문자열(String)

- 다른 값을 문자열로 변환할 때는 String 생성자를 사용하거나 빈 문자열("")과 더하면 된다.

```
String(30) // "30"
"" + 30 // "30"
```

# 연습해봅시다

```
"숫자 " + 7
"숫자 " + 7 * 3
"숫자 " + 7 + 3
```

→ `"숫자 7" + 3`

→ `"숫자 73"`

# 문자열(String)

- 다른 값을 문자열로 변환할 때는 String 생성자를 사용하거나 빈 문자열("")과 더하면 된다.

```
String(30) // "30"
"" + 30 // 30
```

- 문자열을 숫자로 바꿀 때는 `parseInt` 또는 `parseFloat` 함수를 사용하거나 단항 연산자(+, -)를 사용한다.

# 연습해봅시다

```
parseInt("30")
parseInt("30", 16)
parseInt("30a")
parseInt("a30")
parseInt("30.5")
parseFloat("30.5")
+"30"
+"30.5"
+"30a"
-"30"
```

# 문자열(String)

- 다른 값을 문자열로 변환할 때는 `String` 생성자를 사용하거나 빈 문자열(`""`)과 더하면 된다.

```
String(30) // "30"
"" + 30 // 30
```

- 문자열을 숫자로 바꿀 때는 `parseInt` 또는 `parseFloat` 함수를 사용하거나 단항 연산자(`+`, `-`)를 사용한다.
- 활용도가 높은 메서드: `charAt()`, `charCodeAt()`, `replace()`, `indexOf()`, `lastIndexOf()`, `search()`, `slice()`, `split()`, `substr()`, `substring()`, `trim()`, `String.fromCharCode()`
- 활용도가 높은 이스케이프 문자: `\r`, `\n`, `\t`, `\xNN`, `\uNNNN`

# 문자열(String)

- ES6 backtick 문자(`)로 묶으면 템플릿 문자열이 된다.

```
var name = "김태곤";
"안녕하세요, " + name + "님" // "안녕하세요, 김태곤님"
`안녕하세요, ${name}님` // "안녕하세요, 김태곤님"
```

- ES6 템플릿 문자열 안에서 표현식도 사용할 수 있다.

```
var num = 30;
"인생은 " + (num * 2) + "부터" // "인생은 60부터"
`인생은 ${num * 2}부터` // "인생은 60부터"
```

# 불리언(Boolean)

- `true` 또는 `false` 값을 가지는 자료형

# null

- 아무 것도 없이 비어있는 값을 가리킨다.

```
var obj = null; // obj에는 아무 값도 없다.
```

# undefined

- 변수를 선언하고 아무런 값도 할당하지 않을 때의 기본값

```
var name; // undefined
```

- 직접 할당할 수도 있다.

```
var name = undefined;
```

- undefined는 예약어가 아니라서 다른 값을 할당해도 문법 에러가 발생하지 않는다.