◆ JavaScript 구문 (Syntax)

- JavaScript 구문은 JavaScript 프로그램이 구성되는 **규칙의 집합이다.**

예시)

```
// 변수 생성 방법:
var x;
let y;

// 변수 사용 방법:
x = 5;
y = 6;
let z = x + y;
```

◆ 자바스크립트 값 (Values)

- JavaScript 구문은 두 가지 유형의 값 (Values)을 정의한다.
  - 고정 값 : 리터럴이라고 한다.
  - 변수 값 : 변수라고 한다.

◆ 자바스크립트 리터럴 (Literals)

- 고정 값에 대한 두 가지 가장 중요한 구문 규칙은 다음과 같다.
  1. 숫자는 소수점을 포함하거나 포함하지 않고 작성된다. (예시_ 10.50 1001)
  2. 문자열은 큰따옴표나 작은따옴표로 묶인 텍스트이다 (예시_ “홍길동” ‘홍길동’)

작성자: 박정화

◆ 자바스크립트 변수 (Variables)

- 프로그래밍 언어에서 변수는 데이터 값을 저장하는 데 사용된다.
- JavaScript는 **var 또는 let 또는 const 키워드를 사용하여 변수를 선언한다.**
- 등호 (=) 는 변수에 값을 할당하는 데 사용된다.
- 다음 예에서, x는 변수로 정의되며 정의된 변수 x에 6 이라는 (숫자) 값이 할당된다.

```
<p id="demo"></p>

<script>
let x;      //변수 선언
x = 6;      //값 할당
document.getElementById("demo").innerHTML = x;
</script>
```

◆ 자바스크립트 연산자 (Operators)

- JavaScript는 **산술 연산자(+ - * /)**를 사용하여 **값을 계산**한다.

```
<p id="demo"></p>

<script>
document.querySelector("#demo").innerHTML =
(5 + 6) * 10;
</script>
```


작성자: 박정화

- JavaScript는 **할당 연산자(=)**를 사용하여 변수에 **값을 할당**한다.

```
<p id="demo"></p>
<script>
let x, y;
x = 5;
y = 6;
document.getElementById("demo").innerHTML = x + y;
</script>
```

◆ 자바스크립트 표현식 (Expressions)

- 표현식은 값을 계산하는 값, 변수 및 연산자의 조합이다.
- 계산을 평가(evaluation)라고 한다.
- 예를 들어 5 * 10은 50으로 평가된다.
- 표현식에는 변수 값도 포함될 수 있다.

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 * 10;    //표현식으로 계산(평가)이 사용됨
</script>
```

```
<script>
var x;
x = 5;
document.getElementById("demo").innerHTML = x * 10;  //표현식으로 변수와 계산(평가)이 사용됨.
</script>
```

- 값은 숫자 및 문자열과 같은 **다양한 유형일 수 있다**.
- 예를 들어 "홍" + " " + "길동"는 "홍 길동"로 평가된다.

```
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "홍" + " " + "길동";     //"홍 길동"로 평가(계산) 됨.
</script>
```

◆ 자바스크립트 키워드(Keywords)

- JavaScript **키워드는** 수행할 작업을 **식별하는 데 사용**된다.
- **let 키워드**는 브라우저에 **변수를 생성하도록 지시**한다.

```
<p id="demo"></p>
<script>
let x, y;           // let 키워드는 변수를 만든다.
x = 5 + 6;        // 변수 x에는 (계산 후) 11 이라는 숫자가 값으로 대입 됨.
y = x * 10;       // 변수 y에는 (계산 후) 21 이라는 숫자가 값으로 대입 됨.
document.getElementById("demo").innerHTML = y;
</script>
```

- 상황에 따라 다른데, 이번 예제에서는 **var 또는 let을 사용하면 동일한 결과가 생성된다.**

```
<script>
var x, y;           // let 키워드는 변수를 만든다.
x = 5 + 6;        // 변수 x에는 (계산 후) 11 이라는 숫자가 값으로 대입 됨.
y = x * 10;       // 변수 y에는 (계산 후) 21 이라는 숫자가 값으로 대입 됨.
document.getElementById("demo").innerHTML = y;
</script>
```

◆ 자바스크립트 주석 (Comments)

- 모든 JavaScript 문이 반드시 "실행"되는 것은 아니다.
- 이중 슬래시 // 또는 /* ~~ */ 사이의 코드는 주석으로 처리된다.
- 주석은 무시되며 실행되지 않는다.

```
<p id="demo"></p>
<script>
let x;
x = 5;
// x = 6; 이 글자는 실행되지 않는다.
document.getElementById("demo").innerHTML = x;
</script>
```

◆ 자바스크립트 식별자 (Identifiers)

- 식별자는 이름이다.
- JavaScript에서 **식별자는 변수(및 키워드, 함수, 레이블)의 이름을 지정하는 데 사용된다**.
- 법적 이름에 대한 규칙은 대부분의 프로그래밍 언어에서 거의 동일하다.
- JavaScript에서 **첫 번째 문자는 문자, 밑줄(_) 또는 달러 기호($)여야 한다.**
- 후속 문자는 문자, 숫자, 밑줄 또는 달러 기호일 수 있다.
- **숫자는 첫 번째 문자로 사용할 수 없다.**

작성자: 박정화

◆ JavaScript는 대소문자를 구분한다.

- 모든 JavaScript 식별자는 대소문자를 구분한다.
- lastName 및 lastname 변수는 두 개의 다른 변수이다.

```
<p id="demo"></p>
<script>
let lastname, lastName;
lastName = "강 민국";
lastname = "강 대한";
document.getElementById("demo").innerHTML = lastName;
</script>
```

- JavaScript는 LET 또는 Let을 키워드 let으로 해석하지 않는다.

◆ 자바스크립트와 카멜 케이스 (Camel Case)

- 역사적으로 프로그래머는 여러 단어를 하나의 변수 이름으로 결합하는 다양한 방법을 사용했다.
- 하이픈 (Hyphens) : 잘못된 예시) first-name, last-name, master-card, inter-city. X
- 하이픈은 JavaScript에서 허용되지 않는다. 뺄셈을 위해 예약되어 있다.
- 밑줄(Underscore): 예시) first_name, last_name, master_card, inter_city.
- 어퍼 카멜 케이스 (파스칼 케이스): Upper Camel Case (Pascal Case):
  예시) FirstName, LastName, MasterCard, InterCity.

- 로우어 카멜 케이스: (Lower Camel Case:)
- JavaScript 프로그래머는 **소문자로 시작하는** 카멜 케이스를 사용하는 경향이 있다.
  예시) firstName, lastName, masterCard, interCity. O

작성자: 박정화

◆ 자바스크립트 문자 집합 (Character Set)

- JavaScript는 유니코드 문자 집합을 사용한다.
- 유니코드는 (거의) 세계의 모든 문자, 구두점 및 기호를 다룬다.
- 유니코드 컨소시엄
  -유니코드 컨소시엄은 유니코드 표준을 개발합니다. 그들의 목표는 기존 문자 집합을
  표준 UTF(Unicode Transformation Format)로 바꾸는 것이다.
  -유니코드 표준은 성공했으며 HTML, XML, Java, JavaScript, 이메일, ASP, PHP 등으로 구현되었다.
  유니코드 표준은 또한 많은 운영 체제와 모든 최신 브라우저에서 지원된다.
  -Unicode Consortium은 ISO, W3C 및 ECMA와 같은 주요 표준 개발 조직과 협력하고 있다.
  -가장 일반적으로 사용되는 인코딩은 UTF-8 및 UTF-16이다.

◆ HTML5 표준: 유니코드 UTF-8

- ISO-8859의 문자 집합은 크기가 제한되어 있고 다국어 환경에서 호환되지 않기 때문에 유니코드 컨소시엄은 유니코드 표준을 개발했다.
- 유니코드 표준은 세계의 (거의) 모든 문자, 구두점 및 기호를 다룬다.
- 유니코드를 사용하면 플랫폼 및 언어에 관계없이 텍스트를 처리, 저장 및 전송할 수 있다.
- HTML-5의 기본 문자 인코딩은 UTF-8이다.
- HTML5 웹 페이지가 UTF-8과 다른 문자 집합을 사용하는 경우 <meta> 태그에 지정해야 한다.

```
<meta charset="utf-8">
```


작성자: 박정화