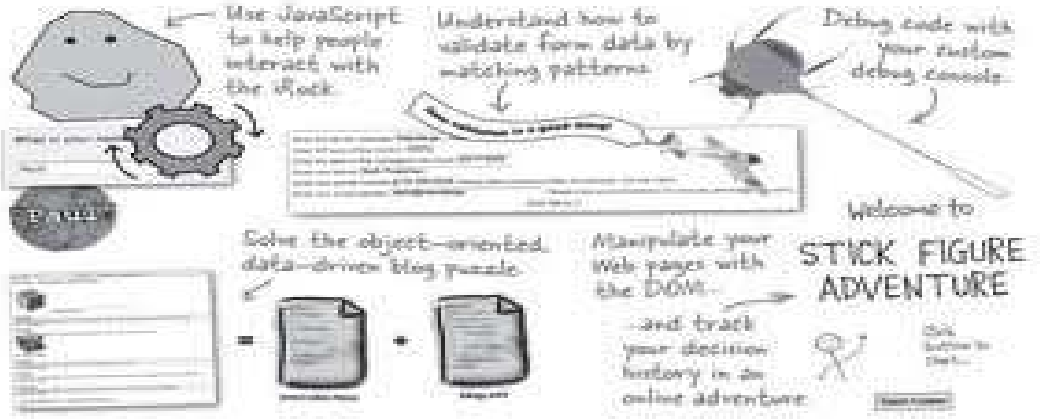
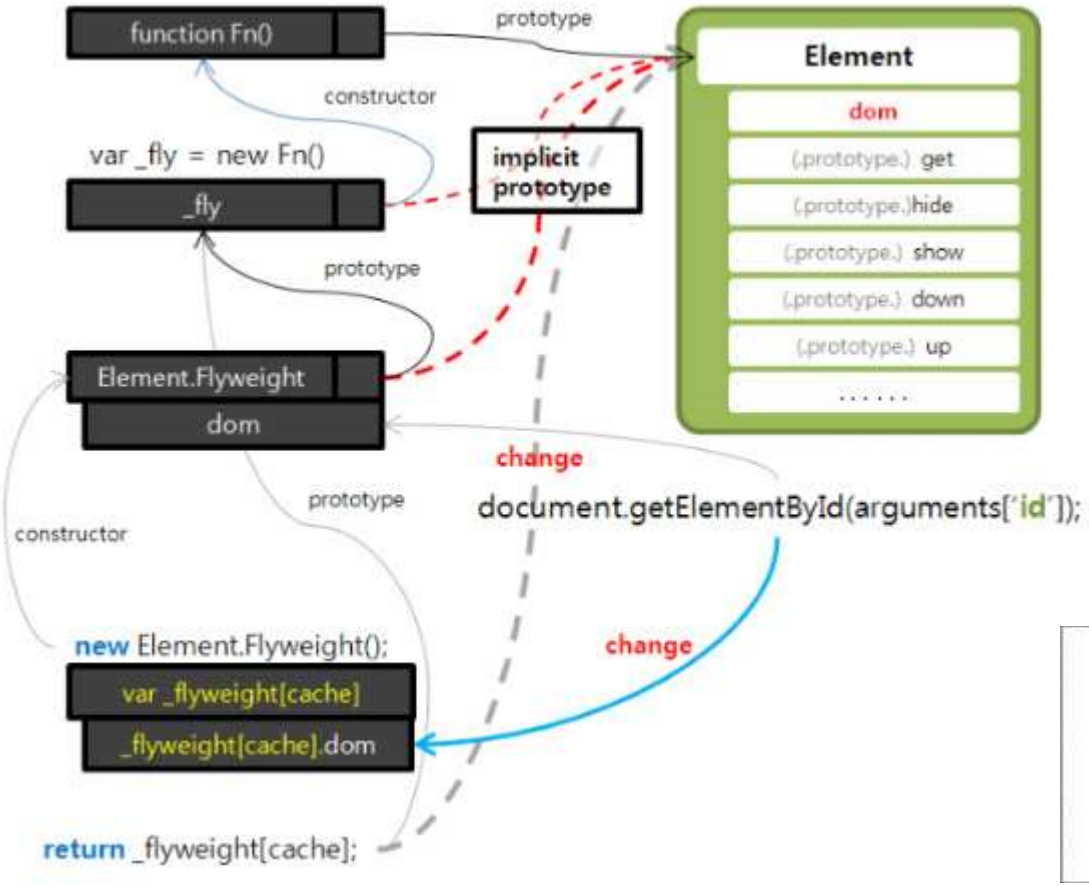
10. 이벤트


Use JavaScript to help people interact with the Rock
Understand how to validate form data by matching patterns
Debug code with your custom debug console
Welcome to
STICK FIGURE ADVENTURE
Solve the object-oriented, data-driven blog puzzle
Manipulate your Web pages with the DOM
and track your decision history in an online adventure


function Fn()
prototype
constructor
var _fly = new Fn()
_fly
implicit prototype
Element
dom
(prototype.) get
(prototype.)hide
(prototype.) show
(prototype.) down
(prototype.) up
.....
prototype
Element.Flyweight
dom
change
prototype
document.getElementById(arguments['id']);
constructor
new Element.Flyweight();
change
var _flyweight[cache]
_flyweight[cache].dom
return _flyweight[cache];


JAVA
SCRIPT

10. 이벤트

❖ 이벤트란?

- 다른 것에 영향을 미치는 것
  - Ex) 키보드로 키를 입력하거나 마우스 클릭
- 자바스크립트가 지원하는 이벤트
  - 애플리케이션 사용자가 발생
  - 애플리케이션이 스스로 발생

  - 마우스 이벤트
  - 키보드 이벤트
  - HTML 프레임 이벤트
  - HTML 입력 양식 이벤트
  - 유저 인터페이스 이벤트
  - 구조 변화 이벤트
  - 터치 이벤트

10.1 이벤트 관련 용어

❖ 이벤트 관련 용어

- "이벤트를 연결한다"
  - Window 객체의 onload 속성에 함수 자료형 할당
- Load - 이벤트 이름 (Event Name) 또는 이벤트 타입 (Event Type)
- onload 이벤트 속성
- 이벤트 핸들러 - 이벤트 속성에 할당한 함수

```
<script>
        window.onload = function () {
            // 변수를 선언합니다.
            var header = document.getElementById('header');
            // 이벤트를 연결합니다.
            function whenClick() { alert('CLICK'); }
            header.onclick = whenClick;
        };
    </script>
```

10. 이벤트 관련 용어

❖ 이벤트 모델의 종류

- DOM Level 0
  - 고전 이벤트 모델
  - 인라인 이벤트 모델
- DOM Level 2
  - 마이크로소프트 인터넷 익스플로러 이벤트 모델
  - 표준 이벤트 모델

10. 고전 이벤트 모델

❖ 고전 이벤트 모델

- 자바스크립트에서 문서 객체의 이벤트 속성 사용 - 이벤트 연결

```
<script>
        window.onload = function () {
            // 변수를 선언합니다.
            var header = document.getElementById('header');
            // 이벤트를 연결합니다.
            header.onclick = function () {
                alert('클릭');
            };
        };
</script>
```

10. 고전 이벤트 모델

❖ 고전 이벤트 모델

- 이벤트 핸들러 제거시 문서 객체의 이벤트 속성에 null 할당
- 이벤트 하나에 이벤트 핸들러 하나

```
<script>
        window.onload = function () {
            // 변수를 선언합니다.
            var header = document.getElementById('header');
            // 이벤트를 연결합니다.
            header.onclick = function () {
                alert('클릭');
                // 이벤트를 제거합니다.
                header.onclick = null;
            };
        };
    </script>
```

10. 이벤트 발생 객체와 이벤트 객체

❖ 이벤트 객체 사용

- 이벤트의 육하원칙 밝힐 수 있음
- this 키워드 – 객체 찾기에 유용

```
<!DOCTYPE html>
<html>
<head>
    <script>
        window.onload = function () {
            document.getElementById('header').onclick = function () {
                alert(this);
            };
        };
    </script>
</head>
<body>
    <h1 id="header">Click</h1>
</body>
</html>
```

10. 이벤트 발생 객체와 이벤트 객체

❖ 이벤트 객체 사용

- 이벤트 핸들러 안에서 this 키워드의 스타일 바꿈
  - 이벤트 발생한 객체의 스타일 변경

```
<!DOCTYPE html>
<html>
<head>
    <script>
        window.onload = function () {
            document.getElementById('header').onclick = function () {
                this.style.color = 'Orange';
                this.style.backgroundColor = 'Red';
            };
        };
    </script>
</head>
<body>
    <h1 id="header">Click</h1>
</body>
</html>
```

10. 이벤트 발생 객체와 이벤트 객체

❖ 이벤트 객체 사용

- 이벤트 객체 내용 출력하는 예제
  - var event = e || window.event;
    - **e가 존재하면 e를 변수 event에 넣고**
    - **e가 undefined이면 window.event 속성을 변수 event에 넣음**
    - **인터넷 익스플로러 8 이하의 버전**
      - » **이벤트가 발생시 이벤트 객체 window.event 속성으로 전달**
      - » **다른 브라우저는 이벤트 핸들러의 매개 변수로 전달**

```
<script>
 window.onload = function () {
    document.body.onclick = function (e) {
    // 이벤트 객체를 설정합니다.
     var event = e || window.event;
     document.body.innerHTML = '';
       for (var key in event) {
         document.body.innerHTML += '<p>' + key + ': ' + event[key] + '</p>';
          }
       };
     };
   </script>
```

10. 이벤트 강제 발생

❖ 이벤트 강제 발생시키는 방법

- 메서드 호출하는 것처럼 이벤트 속성 호출 - 이벤트 강제 실행
  - Ex) header.onclick()
- 연습문제]이벤트 강제 발생
  - 버튼 A 클릭하면 A의 클릭 횟수 1 증가
  - 버튼 B 클릭하면 B의 클릭 횟수는 물론 A의 클릭 횟수까지 증가

코드 10-9 body 태그구성

```
<body>
    <button id="button_a">ButtonA</button>
    <button id="button_b">ButtonB</button>
    <h1>Button A - <span id="counter_a">0</span></h1>
    <h1>Button B - <span id="counter_b">0</span></h1>
</body>
```

10. 인라인 이벤트 모델

❖ 인라인 이벤트 모델

- HTML 페이지의 가장 기본적인 이벤트 연결 방법

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function whenClick(e) {
            alert('클릭');
        }
    </script>
</head>
<body>
    <h1 onclick="whenClick(event)">Click</h1>
</body>
</html>
```

10. 기본 이벤트 제거

❖ 기본 이벤트란?

- 일부 HTML 태그는 이미 이벤트 핸들러 가지고 있음
- 입력양식의 경우 제거하는 경우 발생

```
<script>
        window.onload = function () {
            // 이벤트를 연결합니다.
            document.getElementById('my_form').onsubmit =
function () {
                return false;
            };
        };
/script>
```

10. 기본 이벤트 제거

- 연습문제] 입력 양식의 두개의 패스워드 일치 유효성 검사

```
<body>
    <form id="my_form">
        <label for="name">이름</label><br/>
        <input type="text" name="name" id="name"/><br/>
        <label for="pass">비밀번호</label><br/>
        <input type="password" name="pass" id="pass"/><br/>
        <label for="pass_check">비밀번호 확인</label><br/>
        <input type="password" id="pass_check"/><br/>
        <input type="submit" value="제출"/>
    </form>
</body>
```

10. 기본 이벤트 제거

❖ 디폴트 이벤트 제거

- 고전 이벤트 모델 이벤트 제거시 return false 입력
- 인라인 이벤트 모델을 사용할 때
  - form 태그의 onsubmit 이벤트 속성에 return 함수() 입력

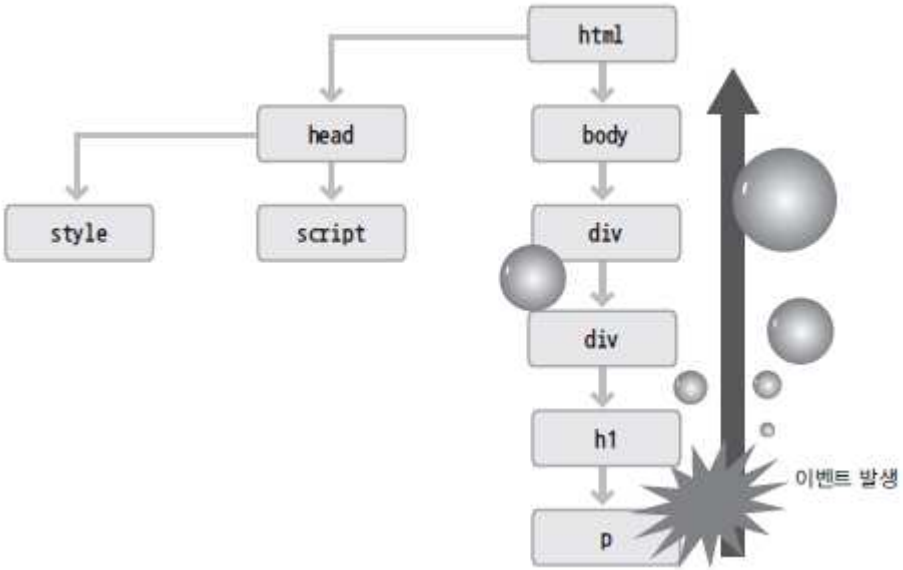
```
<script>
        function whenSubmit() {
            // 변수를 선언합니다.
            var pass = document.getElementById('pass').value;
            var pass_check = document.getElementById('pass_check').value;
            // 비밀번호가 같은지 확인합니다.
            if (pass == pass_check) {
                alert('성공');
            } else {
                alert('다시 입력해주세요.');
                return false;
            }
        }
</script>
```

10. 이벤트 전달

❖ 이벤트 전달

- 이벤트 버블링 방식이 일반적
  - 자식 노드에서 부모 노드 순으로 이벤트 실행

그림 10-8 이벤트 버블링

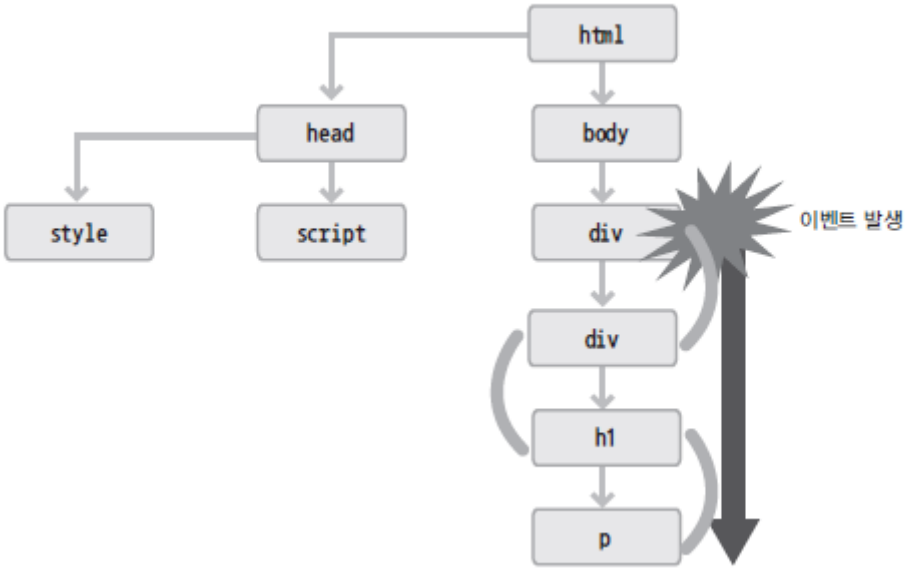


10. 이벤트 전달

❖ 이벤트 전달

- 이벤트 캡쳐링
  - 이벤트가 부모 노드에서 자식 노드 순으로 실행되는 것

그림 10-9 이벤트 캡쳐링



10. 이벤트 전달

❖ 이벤트 전달 막기

- p 태그를 클릭하면 이벤트 버블링
  - paragraph → header 순으로 경고창 출력
- header 경고창을 출력하지 않게 하는 방법
  - 인터넷 익스플로러: 이벤트 객체의 cancelBubble 속성 true로 변경
  - 그 이외의 브라우저: 이벤트 객체의 stopPropagation() 메서드 사용

```
window.onload = function () {
    // 이벤트를 연결합니다.
document.getElementById('header').onclick = function () {
        alert('header');
    };
document.getElementById('paragraph').onclick = function (e) {
        // 이벤트 객체를 처리합니다.
        var event = e || window.event;
        // 이벤트 발생을 알립니다.
        alert('paragraph');
        // 이벤트 전달을 제거합니다.
        event.cancelBubble = true;
        if (event.stopPropagation) {
            event.stopPropagation();
        }
    };
};
```

10. 인터넷 익스플로러 이벤트 모델

❖ DOM Level 2

- 인라인 이벤트 모델, 고전 이벤트 모델의 단점
  - 한 번에 하나의 이벤트 핸들러만 가질 수 있음
- 인터넷 익스플로러 이벤트 모델
  - 두 가지 메서드로 이벤트 연결/제거
  - 첫 번째 매개 변수에 이벤트 속성 !

  - attachEvent(eventProperty,eventHandler);
  - detachEvent(eventProperty,eventHandler);

10.8인터넷 익스플로러 이벤트 모델

❖ 여러 개 이벤트 연결

```
<!DOCTYPE html>
<html>
<head>
    <script>
        // 윈도우가 로드될 때
        window.attachEvent('onload', function () {
            // myHeader를 가져옵니다.
            var header = document.getElementById('myHeader');
            // 이벤트를 연결합니다.
            header.attachEvent('onclick', function () { alert('클릭'); });
            header.attachEvent('onclick', function () { alert('클릭'); });
            header.attachEvent('onclick', function () { alert('클릭'); });
        });
    </script>
</head>
<body>
    <h1 id="myHeader">Click</h1>
</body>
</html>
```

10.8인터넷 익스플로러 이벤트 모델

❖ 이벤트 제거하는 방법

- 익명 함수를 이벤트 핸들러로 사용한 이벤트는 제거할 수 없음
  - detachEvent() 메서드
    - **어떤 이벤트 핸들러를 제거할 지 명확하게 알려주어야 하기 때문**

```
<!DOCTYPE html>
<html>
<head>
    <script>
        window.onload = function () {
            var header = document.getElementById('myHeader');
            var handler = function () { alert('클릭'); };
            header.attachEvent('onclick', handler);
            header.detachEvent('onclick', handler);
        };
    </script>
</head>
<body>
    <h1 id="myHeader">Click</h1>
</body>
</html>
```

10. 인터넷 익스플로러 이벤트 모델

❖ 인터넷 익스플로러 이벤트 모델

- 이벤트 핸들러의 this 키워드는 이벤트 발생 객체 의미하지 않음
  - window 객체 의미
- 이벤트 발생 객체 사용하려면 이벤트 객체의 srcElement 속성 사용
  - attachEvent() 메서드 - 인터넷 익스플로러만 가지고 있음

```
<!DOCTYPE html>
<html><head>
    <script>
        window.onload = function () {
            var header = document.getElementById('myHeader');
            // 인터넷 익스플로러의 경우 실행합니다.
            if (header.attachEvent) {
                var handler = function () {
                    window.event.srcElement.style.color = 'red';
                    window.event.srcElement.detachEvent('onclick', handler);
                };
                header.attachEvent('onclick', handler);
            }
        };
    </script>
</head>
<body>
    <h1 id="myHeader">Click</h1>
</body></html>
```

10. 표준 이벤트 모델

❖ 표준 이벤트 모델

- 웹 표준 단체인 W3C에서 공식 지정한 DOM Level 2 이벤트 모델
- 한 번에 여러 가지 이벤트 핸들러 추가 가능
- 이벤트 연결 메서드
  - 이벤트 이름 매개 변수로 입력 !!!
  - 매개 변수 useCapture는 입력하지 않으면 자동으로 false 입력
    - **addEventListener(eventName, handler, useCapture)**
    - **removeEventListener(eventName, handler)**

10.9 표준 이벤트 모델

❖ 표준 이벤트 모델

- Click 이벤트 연결 예제

```
<!DOCTYPE html>
<html>
<head>
    <script>
        window.onload = function () {
            var header = document.getElementById('myHeader');
            header.addEventListener('click', function () {
                this.innerHTML += '+';
            });
        };
    </script>
</head>
<body>
    <h1 id="myHeader">Click</h1>
</body>
</html>
```

10. 표준 이벤트 모델

❖ 표준 이벤트 모델

- 이벤트 모델의 통합적 사용예제

```
<!DOCTYPE html>
<html><head>    <script>
        window.onload = function () {
            var header = document.getElementById('myHeader');
            if (header.attachEvent) {
                // 인터넷 익스플로러의 경우 실행합니다.
                var handler = function () {
                    window.event.srcElement.style.color = 'red';
                    window.event.srcElement.detachEvent('onclick', handler);
                };
                header.attachEvent('onclick', handler);
            } else {
                // 그 이외의 브라우저에서 실행합니다.
                var handler = function () {
                    this.style.color = 'red';
                    this.removeEventListener('click', handler);
                };
                header.addEventListener('click', handler);
                header.addEventListener('click', handler);
            }
        };
    </script></head>
<body>
    <h1 id="myHeader">Click</h1>
</body></html>
```