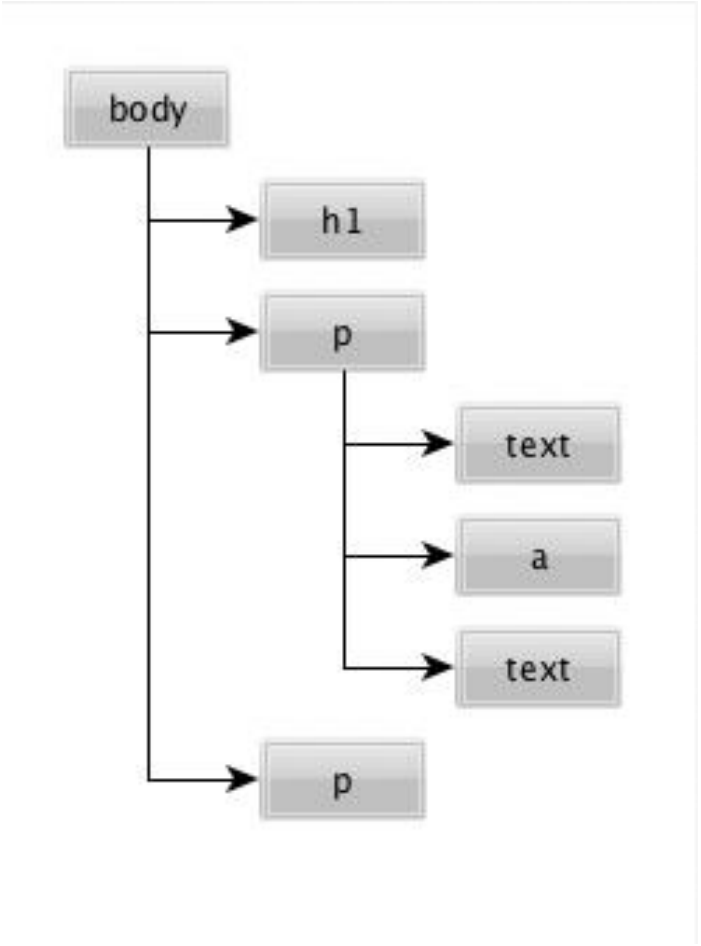
Introduction to the Document Object Model

What is the DOM?

The Document Object Model is what allows web pages to render, respond to user events, and change

The DOM is a tree

- The main idea here: There is a root Node that branches into other Nodes, known as its children Nodes
  - Each Node can have none or many children Nodes
  - Nodes can have 0 or 1 parent
  - Nodes can have 0 to many Sibling Nodes



Why do we care?

The DOM makes it possible to use JavaScript to manipulate the document content and structure

Nodes have lots of Attributes

- Nodes are JavaScript Objects
- Nodes have Attributes that are JavaScript properties
- Attributes define how the Node looks and responds to User activity

The *document* Object

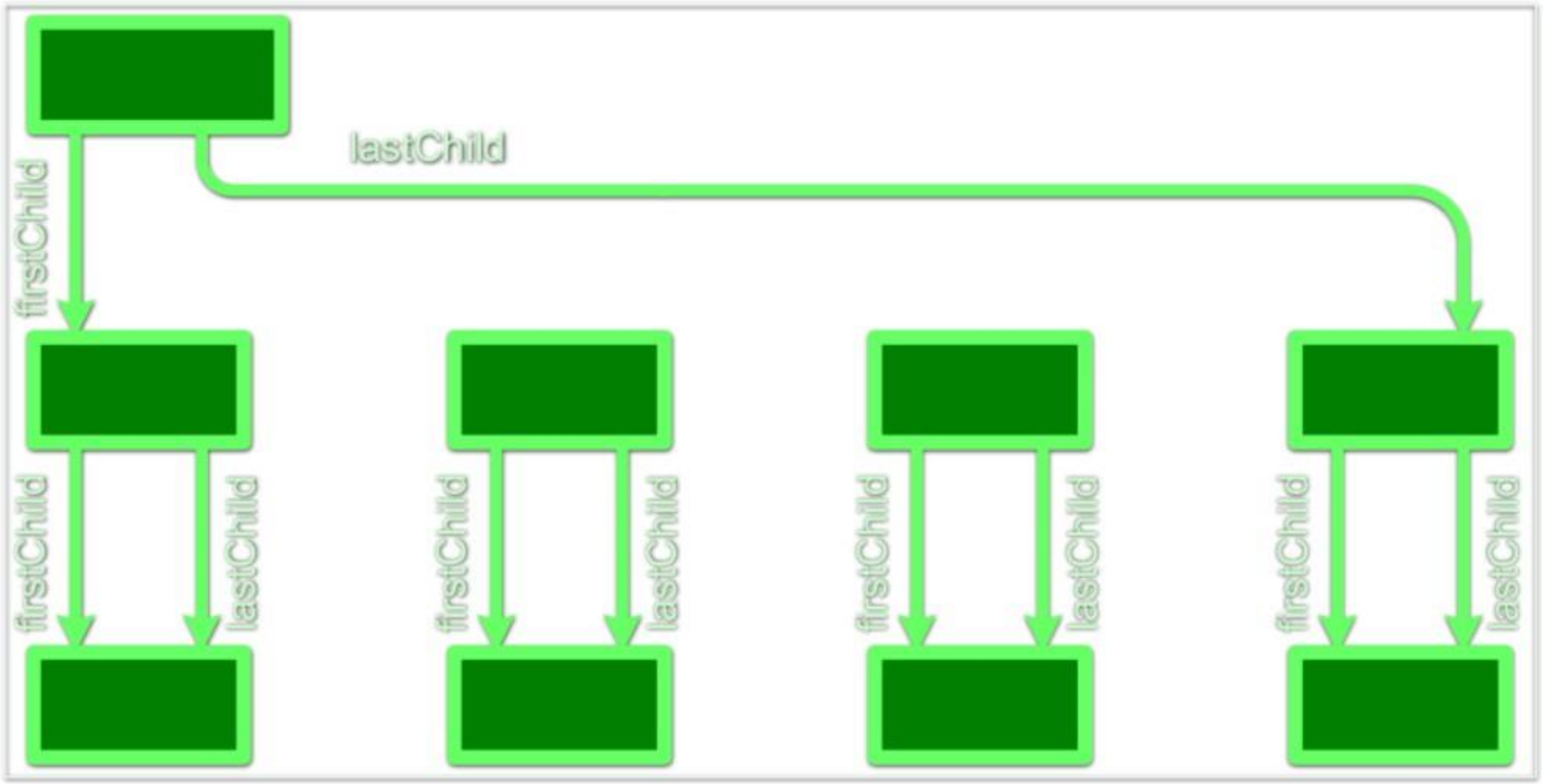
- ‘document’ is the Global reference to the DOM entry point
- Provides methods for navigating and manipulating the DOM
- The *document* object is the important connection between the DOM and JavaScript code˜

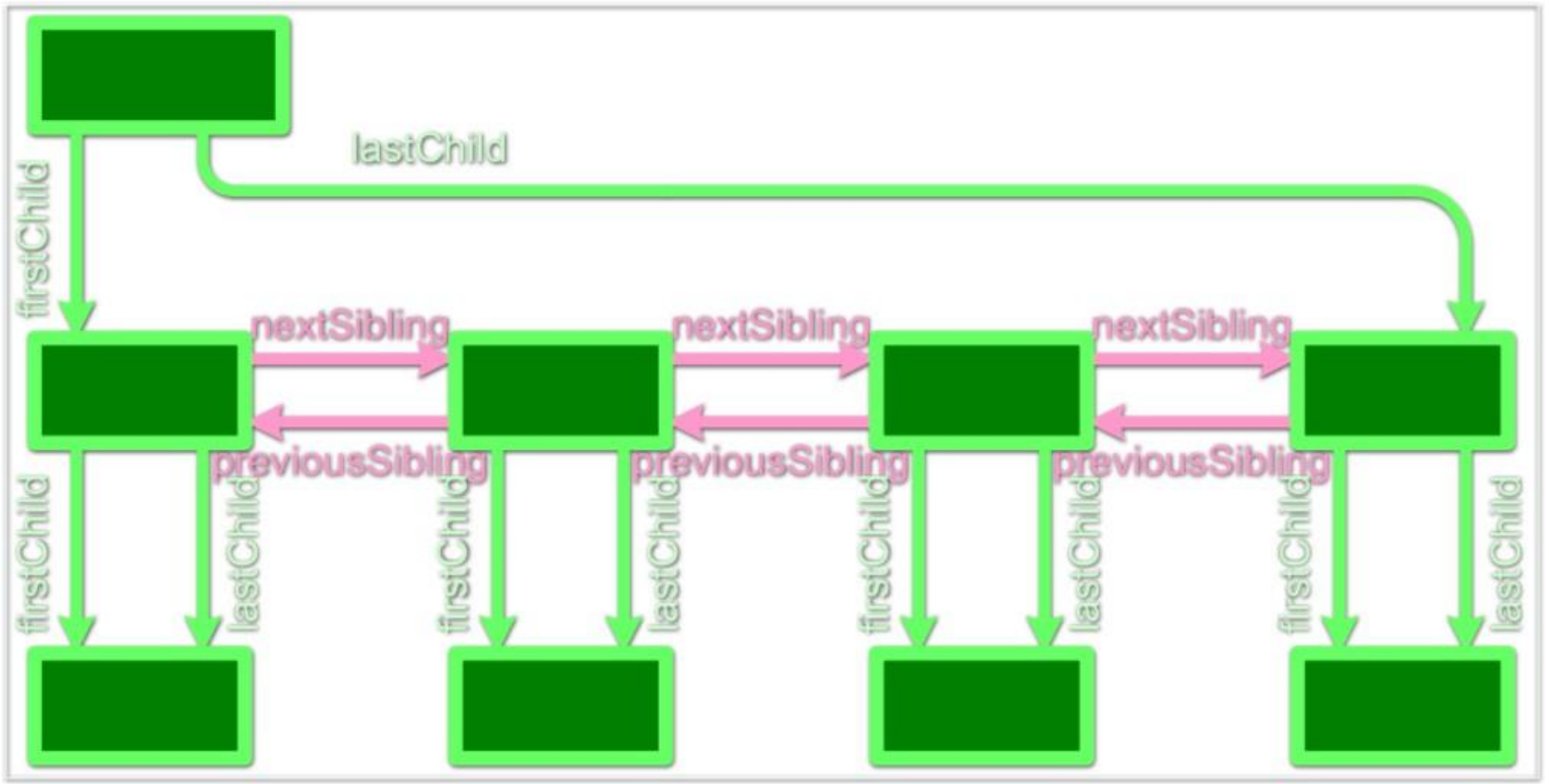
Searching the DOM

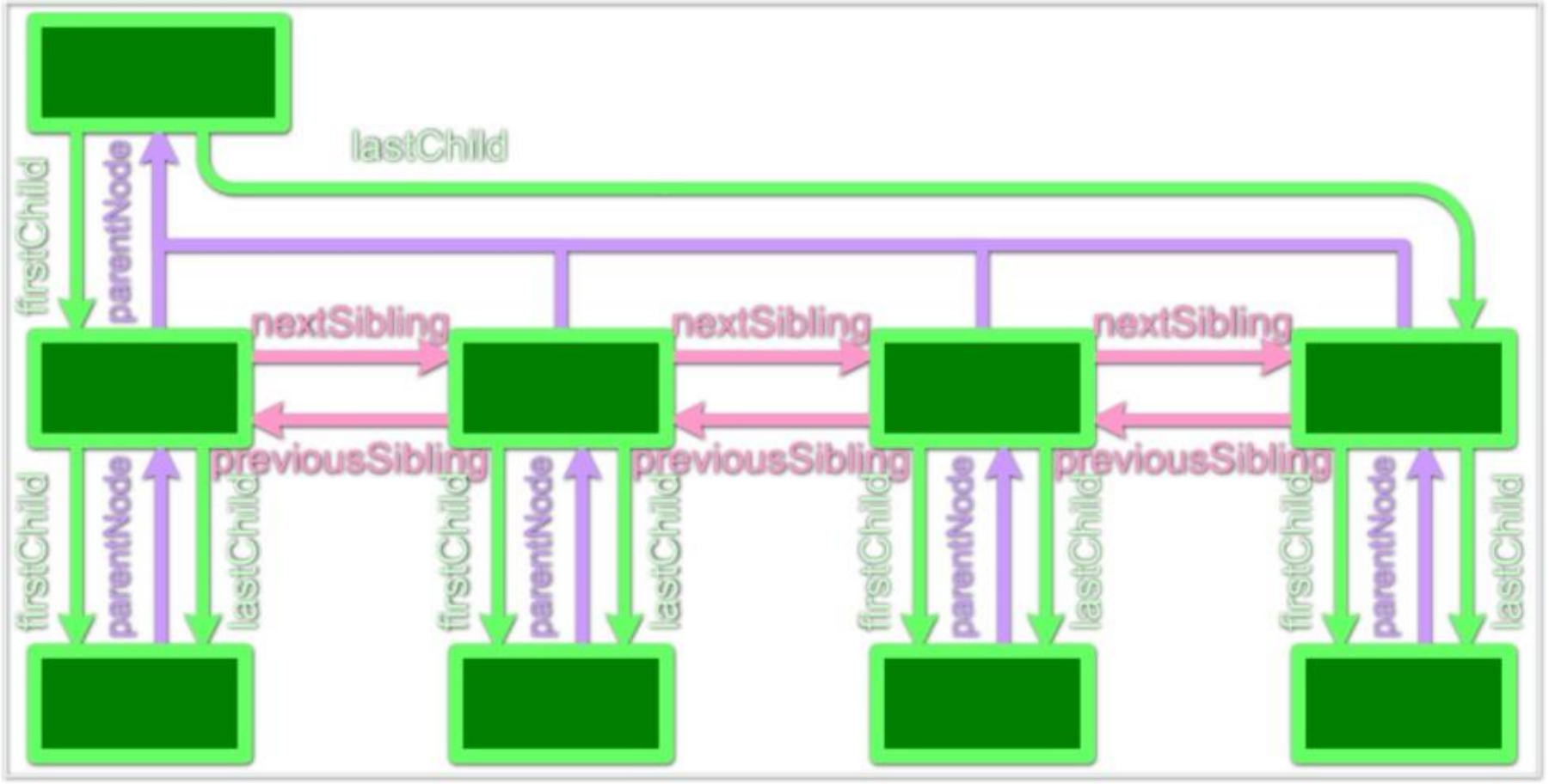
- **getElementById - finds node with a certain ID attribute**
  - document.getElementById(“will”);
- **getElementsByClassName - finds nodes with a certain CLASS attribute**
  - document.getElementsByClassName(“will”)
- **getElementsByTagName - finds nodes with a certain HTML tag**
  - document.getElementsByTagName(“div”);
- **querySelector, querySelectorAll - searches using CSS selectors**
  - document.querySelector(“#will .will:first-child”);

Traversing the DOM

- As the DOM is a Tree Structure, it is relatively easy to navigate because:
  - At any point in the DOM you are at a Node
  - No matter where you go, you're still at a Node
    - Child
    - Parent
    - Sibling
  - All Nodes share similar DOM navigation methods


lastChild
firstChild
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild


lastChild
firstChild
nextSibling
nextSibling
nextSibling
previousSibling
previousSibling
previousSibling
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild
firstChild
lastChild


lastChild
firstChild
parentNode
nextSibling
nextSibling
nextSibling
previousSibling
previousSibling
previousSibling
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild
firstChild
parentNode
lastChild

Traversing the DOM

- Access children Nodes

element.children, element.lastChild, element.firstChild

- Access sibling Nodes

element.nextElementSibling, element.previousElementSibling

- Access parent Node (if any)

element.parentElement

Changing the DOM: Changing style attributes

```
element.style.fontWeight = “bold”;
```

- CSS
  - background-color
  - border-radius
  - font-weight
  - list-style-type
  - word-spacing
  - z-index
- JavaScript
  - backgroundColor
  - borderRadius
  - fontWeight
  - listStyleType
  - wordSpacing
  - zIndex

Changing the DOM: Changing CSS Classes

- *className* attribute is a string of all of a Node's classes
- *classList* is HTML5 way to modify which classes are on a Node

```
document.getElementById("MyElement").classList.add('class');

document.getElementById("MyElement").classList.remove('class');

if(document.getElementById("MyElement").classList.contains('class')){
        document.getElementById("MyElement").classList.toggle('class');
}
```

Changing the DOM: Creating Elements

- Create an element
  - document.createElement(tagName)
- Duplicate an existing Node
  - node.cloneNode()
- Nodes are just free floating, and not connected to the document itself until you link them to the DOM.

Changing the DOM: Adding Elements to the DOM

- Insert newNode at end of current Node
  - node.appendChild(newNode);
- Insert newNode at beginning of current Node
  - node.prependChild(newNode);
- Insert newNode before a certain childNode
  - node.insertBefore(newNode, sibling);

Changing the DOM: Removing Elements

- Remove the oldNode child
  - node.removeChild(oldNode)
- Quick hack:
  - oldNode.parentNode.removeChild(oldNode)

JS Event Handling

What is an event?

- A JavaScript event is a callback that gets fired when something happens related to the DOM on your website.
- For instance, when an element is clicked, or perhaps hovered over
- An event handler can be attached to an element so that when a specific event happens, a specific “callback” function gets fired

Listening for events - native JS

```
document.getElementById("myId").addEventListener("click", function(event){  alert('you clicked me')
}
```

- The key bit to the snippet above is the .addEventListener, which attaches an event handler (an anonymous function to execute) when the element is clicked
- There are many other events to listen for, too, such as:
  - hover
  - keyup / keydown
  - mouseover
  - scroll

The HTML <form>element

- The login, signup, and address forms you see online all share a common tag: <form>
- Inside of <form>are several elements that make up forms:
  - Text input boxes
  - Dropdown
  - Radio buttons,
  - Checkboxes, etc

<form>example

```
<form action="/process" method="POST">
  <label for="username">Username</label>
  <input type="text" name="username" id="username">
  <label for="password">Password</label>
  <input type="password" name="password">
  <input type="submit" value="Submit">
</form>
```

Don’t worry about action and method for now - also don’t worry about submitting your form just yet.

Retrieve input from a form element

You can see what's inside of a form element fairly easily, using the .value attribute:

```
<!-- Sample form input element -->
<input type="text" name="username" id="username">

document.getElementById('username').value
// returns the value of the field
```

Get the title of the form

Imagine a <form> with an <h1> tag above it that has the form title. We can use the attribute .innerText to retrieve the title inside the <h1> tag, or even change it.

Get the title of the form

```
<h1 id="title">Enter your information</h1>


heading = document.getElementById('title')
heading.innerText
>> "Enter your information"

// set name to Zach
var name = "Zach"

// Changes the text in the DOM
heading.innerText = "Enter " + name + "'s Information"

// inside of <h1> to say this instead
heading.innerText
>> "Enter Zach's Information"
```

Change the content of a

Let’s now say that our <h1> lives inside of a . Using the .innerHTML attribute, we can change the innerHTML of the entirely.

Change the content of a

Before:

```
<div id="main-section">
  <h1>Hello World</h1>
</div>
```

JS:

```
document.getElementById('main-section').innerHTML = "<h3>Hello World Smaller</h3>"
```

After:

```
<div class="main-section">
  <h3>Hello World Smaller</h3>
</div>
```

01 Basic terms and concepts of the Document Object Model

- **Document Object Model Basic Terms**
  - Static generation
    - Creating a document object marked with HTML tags the first time you run a web page.
  - Dynamic Generation
    - Using JavaScript to create a document object while running a web page
  - Document Object Model, DOM
    - How web browsers analyze and output HTML files

02 Selecting Document Objects

- **'Select a document object'.**
  - Converting an already existing HTML tag from JavaScript to a document object

표 10-1 문서 객체를 선택하는 메서드

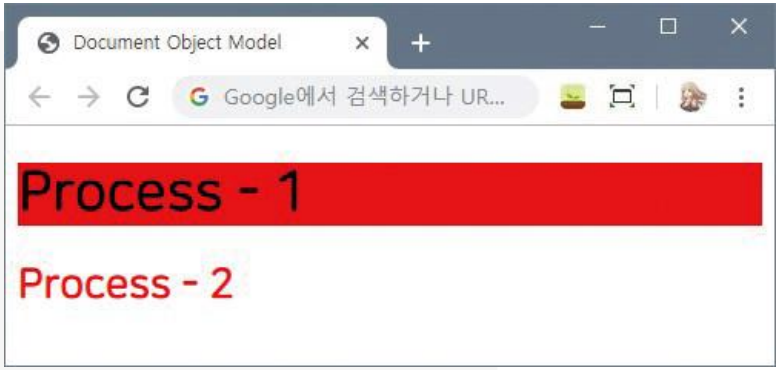
구분	메서드	설명
1개 선택	document.getElementById(아이디)	아이디로 1개 선택
	document.querySelector(선택자)	선택자로 1개 선택
여러 개 선택	document.getElementsByName(이름)	name 속성으로 여러 개 선택
	document.getElementsByClassName(클래스)	class 속성으로 여러 개 선택
	document.querySelectorAll(선택자)	선택자로 여러 개 선택

01 Basic terms and concepts of the Document Object Model

코드 데모

코드 10-3 dom_correct.html

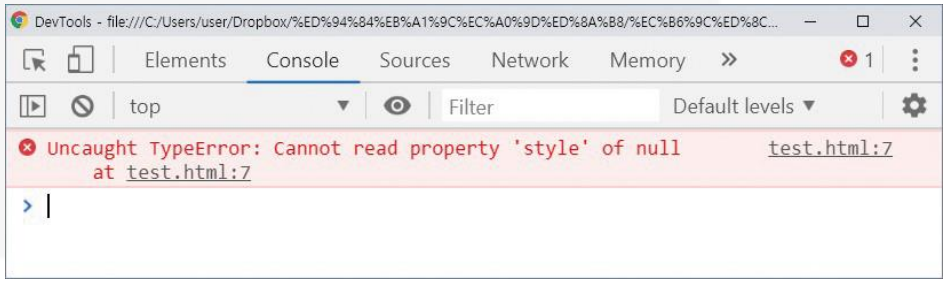
```
<!DOCTYPE html>
<html>
<head>
    <title>Document Object Model</title>
</head>
<body>
    <h1>Process - 1</h1>
    <h2>Process - 2</h2>
    <script>
        // h1 태그의 배경 색상을 변경합니다.
        document.querySelector('h1').style.backgroundColor = 'red';
        // h2 태그의 글자 색상을 변경합니다.
        document.querySelector('h2').style.color = 'red';
    </script>
</body>
</html>
```



- **Error using document objects in execution order**
  - 1. Error using document object

코드 10-2 dom_fault.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Document Object Model</title>
    <script>
        // h1 태그의 배경 색상을 변경합니다.
        document.querySelector('h1').style.backgroundColor = 'red';
        // h2 태그의 글자 색상을 변경합니다.
        document.querySelector('h2').style.color = 'red';
    </script>
</head>
<body>
    <h1>Process - 1</h1>
    <h2>Process - 2</h2>
</body>
</html>
```



01 Basic terms and concepts of the Document Object Model

- **Using events**

코드 10-4 dom_event.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Document Object Model</title>
    <script>
        window.onload = function () {
            // h1 태그의 배경 색상을 변경합니다.
            document.querySelector('h1').style.backgroundColor = 'red';
            // h2 태그의 글자 색상을 변경합니다.
            document.querySelector('h2').style.color = 'red';
        };
    </script>
</head>
<body>
    <h1>Process - 1</h1>
    <h2>Process - 2</h2>
</body>
</html>
```

02 Selecting Document Objects

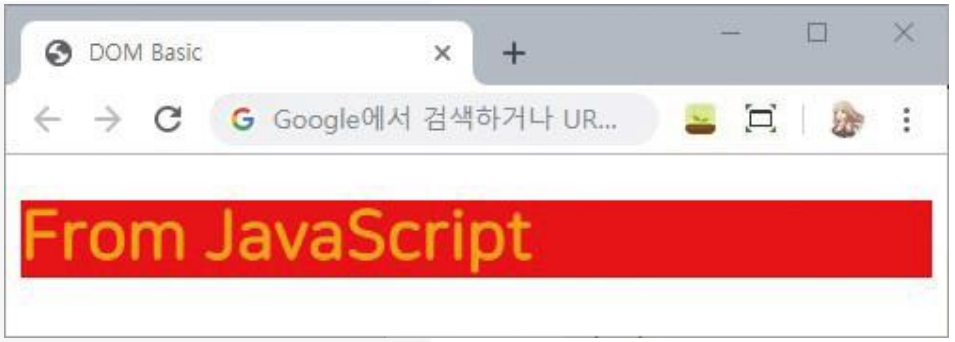
- **Use the getElementById( ) method to select 1 document object**

코드 10-6 select_id.html

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Basic</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 문서 객체를 선택합니다.
            var header = document.getElementById('header');

            // 문서 객체를 조작합니다.
            header.style.color = 'orange';
            header.style.background = 'red';
            header.innerHTML = 'From JavaScript';
        };
    </script>
</head>
<body>
    <h1 id="header">Header</h1>
</body>
</html>
```

태그 내부를 의미하는 속성입니다.



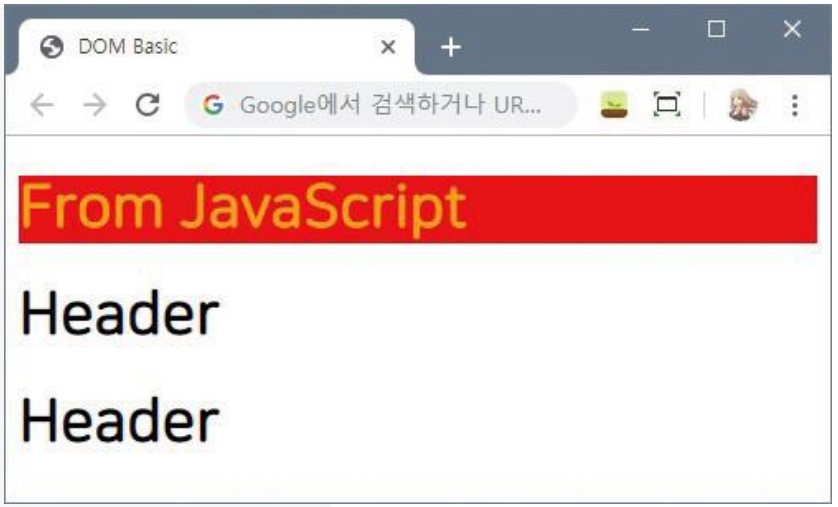
02 Selecting Document Objects

- **Using querySelector( )**

코드 10-7 select_query.html

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Basic</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 문서 객체를 선택합니다.
            var header = document.querySelector('h1');

            // 문서 객체를 조작합니다.
            header.style.color = 'orange';
            header.style.background = 'red';
            header.innerHTML = 'From JavaScript';
        };
    </script>
</head>
<body>
    <h1>Header</h1>
    <h1>Header</h1>
    <h1>Header</h1>
</body>
</html>
```



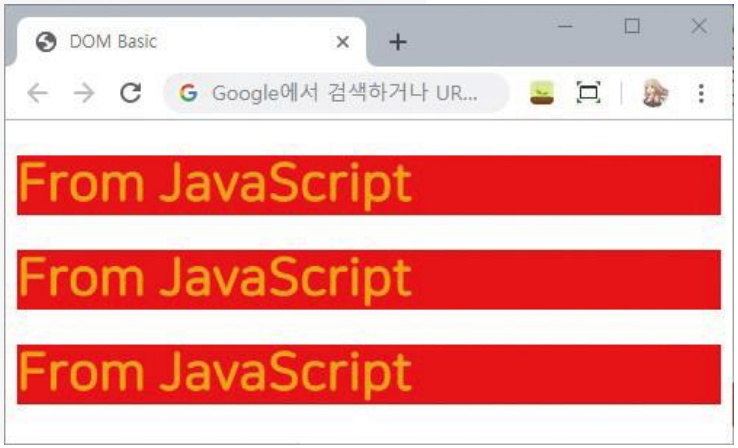
02 Selecting Document Objects

- **Select multiple objects using querySelectorAll( )₩**

코드 10-8 select_all.html

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Basic</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 문서 객체를 선택합니다.
            var headers = document.querySelectorAll('h1');

            for (var i = 0; i < headers.length; i++) {
                // 변수를 선언합니다.
                var header = headers[i];
                // 문서 객체를 조작합니다.
                header.style.color = 'orange';
                header.style.background = 'red';
                header.innerHTML = 'From JavaScript';
            }
        };
    </script>
</head>
<body>
    <h1>Header</h1>
    <h1>Header</h1>
    <h1>Header</h1>
</body>
</html>
```



- **Style manipulation**
  - JavaScript can add, remove, and change CSS attribute value
  - 1.자바스크립트는 특수 문자 '-'을 식별자에 사용할 수 없으므로 오류 출력

```
❶ var header = document.getElementById('header');
  header.style.background-color = 'red';
```

  - 2. - 로 연결된 단어의 첫 글자를 대문자로 변경

```
❷ var header = document.getElementById('header');
  header.style.backgroundColor = 'red';
```

표 10-3 스타일 식별자 변환

스타일시트의 스타일 속성	자바스크립트의 스타일 식별자
background-image	backgroundImage
background-color	backgroundColor
box-sizing	boxSizing
list-style	listStyle

Style Manipulation (1)

코드 10-10 control_style.html

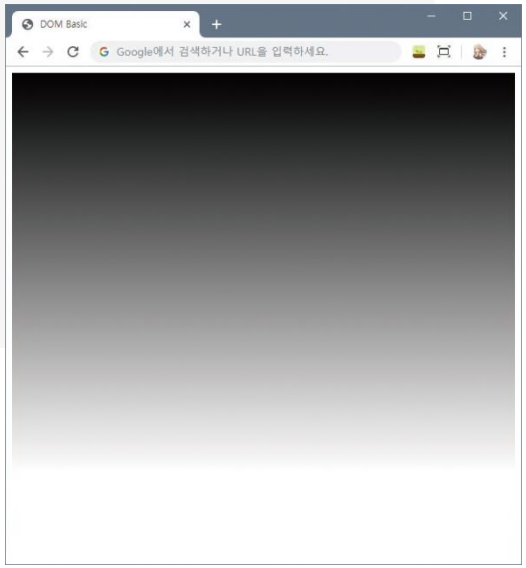
```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Basic</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 문서 객체를 추가합니다.
            var output = '';
            for (var i = 0; i < 256; i++) {
                output += '<div></div>';
            }
            document.body.innerHTML = output;
```

- **Style Manipulation (2)**

```
            // 문서 객체를 선택합니다.
            var divs = document.querySelectorAll('div');
            for (var i = 0; i < divs.length; i++) {
                // 변수를 선언합니다.
                var div = divs[i];

                // 스타일을 적용합니다.
                div.style.height = '2px';
                div.style.background = 'rgb(' + i + ',' + i + ',' + i + ')';
            }
        };
    </script>
</head>
<body>

</body>
</html>
```



- **Attribute manipulation**

- 표 10-4 문서 객체의 속성 조작 메서드

메서드	설명
setAttribute(속성 이름, 속성 값)	속성 지정
getAttribute(속성 이름)	속성 추출

  - setAttribute( ), getAttribute( ) method
    - Available when accessing properties not specified by web standards

```
<body>
    <div data-role="page">
        <div data-role="header"></div>
        <div data-role="content">

        </div>
        <div data-role="footer"></div>
    </div>
</body>
```

그림 10-2 jQuery Mobile 코드

Manipulating Document Object Properties

To Manipulate the img Tag Attribute

코드 10-11 control_attribute.html

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Basic</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 변수를 선언합니다.
            var image = document.getElementById('image');

            // 속성을 변경합니다.
            image.src = 'http://placehold.it/300x200';
            image.width = 300;
            image.height = 200;
        };
    </script>
</head>
<body>
    <img id="image">
</body>
</html>
```

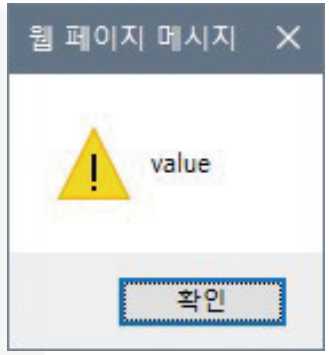
- **Manipulating Document Object Properties**
  - **2. body 태그 속성 조작하기**

코드 10-12 control_body.html

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Basic</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 속성을 지정합니다.
            document.body.setAttribute('data-custom', 'value');

            // 속성을 추출합니다.
            var dataCustom = document.body.getAttribute('data-custom');
            alert(dataCustom);
        };
    </script>
</head>
<body>

</body>
</html>
```




```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
   <body data-custom="value"></body>
</html>
```

그림 10-3 검사로 속성 조작 확인

- **Displaying time using document objects**
  - 2. Manipulate the body tag attribute

코드 10-13 dom_clock.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Clock</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 문서 객체를 선택합니다.
            var clock = document.getElementById('clock');
            // 1초마다 함수를 실행합니다.
            setInterval(function () {
                var now = new Date();
                clock.innerHTML = now.toString();
            }, 1000);
        };
    </script>
</head>
<body>
    <h1 id="clock"></h1>
</body>
</html>
```

- **Event Glossary of Terms**
  - Event property - onload
  - Event name or event type—load except on
  - Event listener or event handler—A function that you put in an event property.

- **Natively supported events**
  - Mouse Events
  - Keyboard Events
  - HTML Frame Events
  - HTML Input Form Events
  - User Interface Events
  - Structural Change Events
  - Touch Events

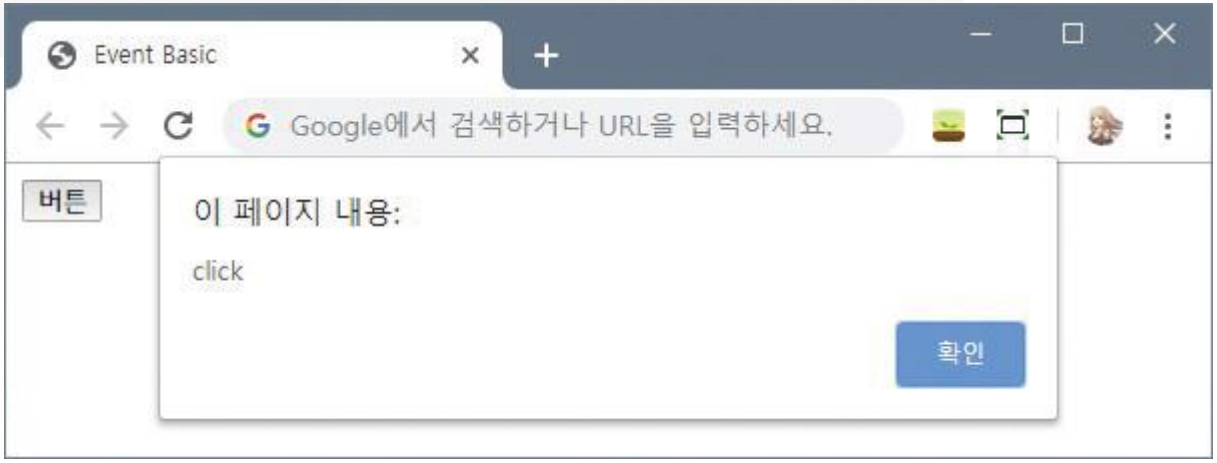
```
onblur  onfocus  onfocusin  onfocusout
onload  onresize  onscroll  onunload
onclick  ondbclick  onmousedown
onmouseup  onmousemove  onmouseover
onmouseout  onmouseenter
onmouseleave  onchange  onselect
onsubmit  onkeydown  onkeypress
onkeyup  onerror
```

그림 10-4 이벤트 속성 종류와 통합 개발 환경의 속성 자동 완성

- **Associate an Event**
- **1. Using the Inline Event Model**

코드 10-14 event_inline.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Event Basic</title>
</head>
<body>
    <button onclick="alert('click')">버튼</button>
</body>
</html>
```



- **Associate an Event**
  - 1. Using the Inline Event Model in Script Tags

코드 10-15 event_inlineWithScript.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Inline Event</title>
    <script>
        function buttonClick() {
            alert('click');
        }
    </script>
</head>
<body>
    <button onclick="buttonClick()">버튼</button>
</body>
</html>
```

- **Associate an Event**
  - Classic Event Model - An event model that was defined as a standard in the past and used a lot.
- **Linking Classic Events**

코드 10-16 event_tradition.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Traditional Event</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 문서 객체를 선택합니다.
            var button = document.getElementById('button');
            // 이벤트를 연결합니다.
            button.onclick = function () {
                alert('click');
            };
        };
    </script>
</head>
<body>
    <button id="button">버튼</button>
</body>
</html>
```

이벤트 리스너

Remove the default event

코드 10-19 event_default.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Traditional Event - Default Event</title>
    <script>
        // 이벤트를 연결합니다.
        window.onload = function () {
            // 문서 객체를 선택합니다.
            var button = document.getElementById('button');

            // 이벤트를 연결합니다.
            button.onclick = function () {
                // 기본 이벤트를 제거합니다.
                return false;
            };
        };
    </script>
</head>
<body>
    <a id="button" href="http://hanbit.co.kr">버튼</a>
</body>
</html>
```