정답 및 해설

1-1 자바스크립트의 활용

1. 직접 다양한 사이트를 찾아보세요. 언뜻 떠오르지 않는다면 유명한 IT 기업들의 사이트를 먼저 확인해보세요. 유튜브, 페이스북, 트위터, 핀터레스트 등도 살펴보면 좋습니다.

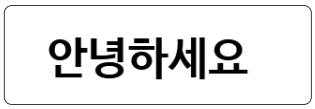
2. StatCounter 사이트에 들어가면 현재 시점의 웹 브라우저 점유율을 확인할 수 있습니다. 이외에 모바일 브라우저까지 합친 통계도 살펴보시기 바랍니다.

1-2 개발환경 설치와 코드 실행

1. [Console] 탭에서 간단하게 입력해보면 답을 알 수 있습니다. 마지막 코드는 오류를 발생합니다. 어떻게 수정해야 할까요? 따옴표가 닫히지 않았으므로 따옴표를 닫아주면 됩니다.

```
> "안녕하세요"
"안녕하세요"
> console.log("안녕하세요")
안녕하세요
undefined
> "안녕하세요
VM244:1 Uncaught SyntaxError: Invalid or unexpected token
```

2. 화면에 큰 글씨로 "안녕하세요"가 출력됩니다.



1-3 알아두어야 할 기본 용어

1. 답은 다음과 같습니다. ④와 ⑤도 사용할 수 있다는 것을 꼭 기억해두세요.

① O　② O　③ X → 숫자로 시작하는 식별자는 사용할 수 없습니다.

④ O　⑤ O

2. ② console.log()에서 console은 식별자입니다. 이후에 내용을 배우면 알 수 있겠지만, 식별자 〉 변수 〉 객체로 구분할 수 있습니다.

3. ②, ④ console.log()에서 log는 식별자이면서 괄호가 있으므로 메소드입니다.

4. ① weAreTheWorld ② createOutput ③ createRequest ④ initServer ⑤ initMatrix

5. ① console을 konsole 등으로 잘못 입력했을 경우 발생할 수 있는 오류입니다.

```
Uncaught ReferenceError: konsole is not defined
```

② +++이라는 연산자가 존재하지 않으므로 오류를 발생합니다.

```
Uncaught SyntaxError: Invalid left-hand side expression in postfix operation
```

③ 괄호를 열고서 닫지 않아서 발생하는 오류입니다.

```
Uncaught SyntaxError: Unexpected token ')'
```

02-1 기본 자료형

1.

&&	불
-	숫자
*	숫자
||	불

2. "\\\\\\\\"는 이스케이프 문자로 처리되므로 \\\\가 4개만 출력됩니다.

```
# 연습문제
\\\\
```

3. 인덱스가 0부터 시작한다는 것을 꼭 기억해주세요!

```
녕
하
세
요
```

4. 연산자 우선 순위가 헷갈리는 문제입니다. 이렇게까지 복잡해진다면 꼭 괄호를 써서 누구나 쉽게 읽을 수 있게 해주세요.

```
0
4
```

02-2 상수와 변수

1. ① 상수를 선언할 때는 const 키워드를 사용합니다. let 키워드는 변수를 선언할 때 사용합니다.

2. ② 값을 할당할 때는 = 연산자를 사용합니다. ①은 과거에 사용되던 프로그래밍 언어에서 많이 사용되던 할당 연산자이며, ③과 ④는 비교 연산자입니다.

3. ① const 키워드를 사용해서 만드는 상수는 반드시 생성할 때 초기화를 해야 합니다. 앞의 2줄을 const r = 10으로 수정하면 정상적으로 동작합니다.

```
Uncaught SyntaxError: Missing initializer in const declaration
```

4. 오류를 발생합니다. ++ 연산자와 -- 연산자는 += 1, -=1과 같은 연산자입니다. 내부적으로 할당하므로 상수에는 적용할 수 없습니다. 초기에 자주 실수할 수 있는 부분이므로 꼭 기억해주세요!

```
Uncaught TypeError: Assignment to constant variable
```

02-3 자료형 변환

1. ③ 불로 입력받을 때는 confirm() 함수를 사용합니다.

2.

String()	문자열 자료형으로 변환합니다.
Boolean()	불 자료형으로 변환합니다.

정답 및 해설

3.

```
<script>
  // 숫자를 입력받습니다.
  const rawInput = prompt('cm 단위의 숫자를 입력해주세요.')

  // 입력을 숫자로 변경하고 cm 단위로 변경합니다.
  const cm = Number(rawInput)
  const inch = cm * 0.393701

  // 출력합니다.
  alert(`${cm}cm는 ${inch}inch 입니다.`)
</script>
```

4.

```
<script>
  // 숫자를 입력받습니다.
  const rawInput = prompt('원의 반지름을 입력해주세요.')
  const r = Number(rawInput)

  // 출력합니다.

  let output = ''
  output += `원의 반지름: ${r}\n`
  output += `원의 넓이: ${3.14 * r * r}\n`
  output += `원의 둘레: ${2 * 3.14 * r}`
  alert(output)
</script>
```

${} 내부에는 표현식을 사용할 수 있으므로 다음과 같이 계산식을 넣을 수도 있습니다. 여러 줄을 좀 쉽게 볼 수 있게 output을 만들고 연결하는 형태로 코드를 작성했습니다.

5.

```
<script>
  // 숫자를 입력받습니다.
  const rawInput = prompt('달러 단위의 금액을 입력해주세요.')
  const dollar = Number(rawInput)

  // 출력합니다.
  let output = ''
  output += `달러: ${dollar}\n`
  output += `→ 원화: ${dollar * 1207}`
  alert(output)
</script>
```

6. 다양한 프로그램을 만들 수 있습니다. 여러분의 상상력을 키울 수 있는 예시를 들어보겠습니다.

① 입력을 하나만 받을 필요도 없습니다. "오늘 아침 점심 저녁 식사의 칼로리로 입력을 3개 받고, 모두 더해서 출력하는 프로그램"도 만들 수 있습니다. **입력을 여러 개 받아보세요!**

② 다큐멘터리에서 "어떤 은하까지 빛의 속도로 여행하면 10년이 걸린다"라고 할 때, 그 거리를 구할 수 있습니다. 빛의 속도 c는 299,792,458m/s입니다. 이를 활용해 10년 * 365일 * 24시간 * 60분 * 60초 * c를 계산하면 거리를 알 수 있습니다. **다양한 공식들을 적용해보세요.**

③ "야! 그걸 왜 사먹냐? 그 돈이면 뜨끈하고 든든한 국밥 n그릇은 사먹을 수 있겠다"라는 유행어가 있었습니다. 이 말처럼 "어떤 돈이 있을 때, 그 돈으로 국밥을 몇 그릇 먹을 수 있는가?"를 구하는 프로그램을 만들어볼 수 있습니다. **유머도 프로그램으로 구현해보세요!**

03-1 if 조건문

1. ③ if 조건문 내부의 불 표현식이 참일 때 if 조건문 내부로 들어갑니다.

2.

```
<script>
  const a = Number(prompt('첫 번째 숫자', ''))
  const b = Number(prompt('두 번째 숫자', ''))

  if (a > b) {
    alert('첫 번째로 입력한 숫자가 더 큽니다.')
  } else if (a === b) {
    alert('두 숫자가 같습니다.')
  } else {
    alert('두 번째로 입력한 숫자가 더 큽니다.')
  }
</script>
```

3.

```
if (x > 10 && x < 20) {
  console.log('조건에 맞습니다.')
}
```

정답 및 해설

4.

```
<script>
  const a = Number(prompt('숫자를 입력해주세요.', ''))

  if (a > 0) {
    alert('입력한 숫자는 양수입니다.')
  } else if (a == 0) {
    alert('입력한 숫자는 0입니다.')
  } else {
    alert('입력한 숫자는 음수입니다.')
  }
</script>
```

5.

```
<script>
  const a = Number(prompt('숫자를 입력해주세요.', ''))

  if (a % 2 === 0) {
    alert('입력한 숫자는 짝수입니다.')
  } else {
    alert('입력한 숫자는 홀수입니다.')
  }
</script>
```

6.

```
<script>
  const a = Number(prompt('월을 입력해주세요.', ''))

  if (3 <= a && a <= 5) {
    alert('봄입니다.')
  } else if (6 <= a && a <= 8) {
    alert('여름입니다.')
  } else if (9 <= a && a <= 11) {
    alert('가을입니다.')
  } else {
    alert('겨울입니다.')
  }
</script>
```

03-2 switch 조건문과 짧은 조건문

1. 100 〉 200는 false이므로 confirm() 함수 부분이 실행되어 "버튼을 클릭해주세요"라는 [확인]과 [취소] 버튼을 클릭할 수 있는 창이 나타납니다. [확인] 버튼을 클릭하면 true, [취소] 버튼을 클릭하면 false가 출력됩니다.

2.

```
<script>
  const rawInput = prompt('태어난 해를 입력해주세요.', '')
  const year = Number(rawInput)
  const e = year % 12

  let result
  switch (e) {
    case 0:
      result = '원숭이'
      break
    case 1:  result = '닭'; break;
    case 2:  result = '개'; break;
    case 3:  result = '돼지'; break;
    case 4:  result = '쥐'; break;
    case 5:  result = '소'; break;
    case 6:  result = '호랑이'; break;
    case 7:  result = '토끼'; break;
    case 8:  result = '용'; break;
    case 9:  result = '뱀'; break;
    case 10: result = '말'; break;
    case 11: result = '양'; break;
  }
  alert(`${year}년에 태어났다면 ${result} 띠입니다.`)
</script>
```

자바스크립트는 문장의 끝을 줄바꿈 또는 세미콜론으로 나타냅니다. 여러 줄로 입력한다면 '원숭이' 부분처럼 할 수 있겠지만, 한 줄에 입력하려고 한다면 세미콜론을 사용해 다음과 같이 입력합니다.

3. 코드가 굉장히 길게 느껴질 수 있지만, 한 번 입력해 보는 것을 추천합니다. 개발의 많은 부분은 이와 같은 속칭 "노가다"로 이루어집니다. "노가다"를 해봐야 이후에 "노가다"를 하지 않는 방법을 깨달았을 때, 그것이 더 유용하게 느껴져서 쉽게 기억할 수 있습니다.

```
<script>
  const rawInput = prompt('태어난 해를 입력해주세요.', '')
  const year = Number(rawInput)

  let 간
  let e = year % 10
  if      (e === 0)  { 간 = '경' }
  else if (e === 1)  { 간 = '신' }
  else if (e === 2)  { 간 = '임' }
  else if (e === 3)  { 간 = '계' }
  else if (e === 4)  { 간 = '갑' }
  else if (e === 5)  { 간 = '을' }
  else if (e === 6)  { 간 = '병' }
  else if (e === 7)  { 간 = '정' }
  else if (e === 8)  { 간 = '무' }
  else if (e === 9)  { 간 = '기' }

  let 띠
  let tti = year % 12
  if      (tti === 0)  { 띠 = '신' }
  else if (tti === 1)  { 띠 = '유' }
  else if (tti === 2)  { 띠 = '술' }
  else if (tti === 3)  { 띠 = '해' }
  else if (tti === 4)  { 띠 = '자' }
  else if (tti === 5)  { 띠 = '축' }
  else if (tti === 6)  { 띠 = '인' }
  else if (tti === 7)  { 띠 = '묘' }
  else if (tti === 8)  { 띠 = '진' }
  else if (tti === 9)  { 띠 = '사' }
  else if (tti === 10) { 띠 = '오' }
  else if (tti === 11) { 띠 = '미' }

  alert(`${year}년은 ${간}${띠} 년입니다.`)
</script>
```

4. ④ **5.** ④

04-1 배열

1. ① “3” ② “바나나” ③ 32

배열은 인덱스가 0부터 시작한다는 것을 꼭 기억하세요!

2. array.push(5)를 하면 [1, 2, 3, 4, 5]를 출력한다고 생각할 수 있지만, 그렇지 않고 그냥 추가된 값을 출력합니다. "사과를 사과라고 하는 이유는 그냥 그렇게 약속해서 그렇다"라고 할 수밖에 없는 것처럼 "그냥 그렇게 설계되어 있다"라고 답할 수밖에 없는 내용입니다.

```
4
5
```

3. ① 비파괴적: strA의 내용이 바뀌지 않았습니다.

② 파괴적: arrayB의 내용이 바뀌었습니다.

③ 비파괴적: arrayC의 내용이 바뀌지 않았습니다.

④ 비파괴적: strD의 내용이 바뀌지 않았습니다.

04-2 반복문

1. for in 반복문과 for of 반복문을 꼭 구분해주세요!

```
# for in 반복문
0
1
2
3
# for of 반복문
사과
배
귤
바나나
```

2. 오류가 발생합니다. for 반복문의 반복 변수는 for in, for of 반복문과 다르게 let 키워드로 변수로서 선언해야 합니다. 실행되게 수정한다면 다음과 같습니다.

```
<script>
  const array = []
  for (let i = 0; i < 3; i++) {
    array.push((i + 1) * 3)
  }
  console.log(array)
</script>
```

3. 초기에 output을 1로 초기화했다는 것에 주의해주세요. 임의의 수 a와 어떤 수를 연산했을 때 a가 나오게 하는 그 어떤 수를 항등원(Identity)이라고 합니다(중학교 수학 과정입니다). 어떤 대상에 여러 번 처리하는 코드를 작성할 때는 이러한 항등원을 꼭 생각해주세요(예를 들어 배열은 항등원이 단위 행렬 등).

<1쇄 내용 오류>
해당 문제의 코드를 다음과 같이 변경합니다.

```
<script>
  let output = 1
  for (let i = 1; i <= 100; i++) {
    output *= i
  }
  console.log(`1~100의 숫자를 모두 곱하면, ${output}입니다.`)
</script>
```

```
<script>
  let output = 0
  for (let i = 1; i <= 100; i++) {
    output += i
  }
  console.log(`1~100의 숫자를 모두 곱하면, ${output}입니다.`)
</script>
```

4. 186쪽 누적 예제의 반복문 형태를 2번 사용하면 됩니다. 물론 절댓값 등을 사용하면 더 간단하게 코드를 작성할 수 있습니다. 여기서는 반복문을 여러 개 사용하는 방법도 있다는 것을 알고 넘어갔으면 하는 바람으로 출제한 문제입니다.

```
<script>
  // 변수를 선언합니다.
  let output = ''
  const size = 5

  // 반복합니다.
  for (let i = 1; i <= size; i++) {
    for (let j = size; j > i; j--) {
      output += ' '
    }
    for (let k = 0; k < 2 * i - 1; k++) {
      output += '*'
    }
    output += '\n'
  }

  for (let i = size -1; i > 0; i--) {
    for (let j = size; j > i; j--) {
      output += ' '
    }
    for (let k = 0; k < 2 * i - 1; k++) {
      output += '*'
```

```
    }
    output += '\n'
  }

  // 출력합니다.
  console.log(output)
</script>
```

05-1 함수의 기본 형태

1. 04-2의 확인 문제 3번을 응용하면 쉽게 만들 수 있습니다.

```
<script>
  const multiplyAll = function (a, b) {
    let output = 0
    for (let i = 1; i <= 100; i++) {
      output += i
    }
    return output
  }

  console.log(multiplyAll(1, 2))
  console.log(multiplyAll(1, 3))
</script>
```

2.

①

```
<script>
  const max = function (array) {
    let output = array[0]
    for (const data of array) {
      if (output < data) {
        output = data
      }
    }
    return output
  }

  console.log(max([1, 2, 3, 4]))
</script>
```

②

```
<script>
  const max = function (...array) {
    let output = array[0]
    for (const data of array) {
      if (output < data) {
        output = data
      }
    }
    return output
  }

  console.log(max(1, 2, 3, 4))
</script>
```

③

```
<script>
  const max = function (first, ...rests) {
    let output
    let items

    if (Array.isArray(first)) {
      output = first[0]
      items = first
    } else if (typeof(first) === 'number') {
      output = first
      items = rests
    }
    for (const data of items) {
      if (output < data) {
        output = data
      }
    }
    return output
  }

  console.log(`max(배열): ${max([1,2,3,4])}`)
  console.log(`max(숫자, ...): ${max(1,2,3,4)}`)
</script>
```

05-2 함수 고급

1.

```
<script>
  // 변수를 선언합니다.
  let numbers = [273, 25, 75, 52, 103, 32, 57, 24, 76]

  // 처리합니다.
  // (1) 홀수만 추출
  numbers = numbers.filter((x) => x % 2 === 1)
  // (2) 100 이하의 수만 추출
  numbers = numbers.filter((x) => x <= 100)
  // (3) 5로 나눈 나머지가 0인 수만 추출
  numbers = numbers.filter((x) => x % 5 === 0)

  // 출력합니다.
  console.log(numbers)
</script>
```

실행 결과

```
[25, 75]
```

2.

```
<script>
  const array = ['사과', '배', '귤', '바나나']

  console.log('# for in 반복문')
  array.forEach((item, i) => {
    console.log(i)
  })

  console.log('# for of 반복문')
  array.forEach((item, i) => {
    console.log(item)
  })
</script>
```

06-1 객체의 기본

1. 처음 코드를 작성할 때 쉼표를 빼먹는 실수를 많이 하므로 빼먹지 않도록 주의하세요!

```
<script>
  const object = {
    name: "혼자 공부하는 파이썬",
    price: 18000,
    publisher: "한빛미디어"
  }
</script>
```

2. ③ **3.** ①

4. print 메소드에 동그라미를 치면 됩니다.

06-2 객체의 속성과 메소드 사용하기

1. 오류 메시지가 나옵니다. num은 기본 자료형이므로 메소드가 추가되지 않습니다. num과 같은 기본 자료형은 대응되는 객체 자료형의 프로토타입(prototype 속성)에 메소드를 추가해야 합니다.

```
Uncaught TypeError: num.원 is not a function
```

2.

```
printLang("ko"): 한국어
printLang("en"): 영어
```

3.

```
<script>
  // 변수를 선언합니다.
  const degree = 90

  // 출력합니다.
  // Math.sin() 메소드는 괄호 내부에 라디안 단위를 넣어야 합니다.
  const radian = degree * (Math.PI / 180)
  console.log(Math.sin(radian))
</script>
```

4. ③

5.

```
<script src="https://cdn.jsdelivr.net/npm/lodash@4.17.15/lodash.min.js">
</script>
<script>
  const books = [{
    name: '혼자 공부하는 파이썬',
    price: 18000,
    publisher: '한빛미디어'
  }, {
    name: 'HTML5 웹 프로그래밍 입문',
    price: 26000,
    publisher: '한빛아카데미'
  }, {
    name: '머신러닝 딥러닝 실전 개발 입문',
    price: 30000,
    publisher: '위키북스'
  }, {
    name: '딥러닝을 위한 수학',
    price: 25000,
    publisher: '위키북스'
  }]

  console.log(_.orderBy(books, (book) => book.name))
</script>
```

06-3 객체와 고급 배열

1. ②

2. 리액트(React), 뷰(Vue), 제이쿼리(jQuery), Luxon, Day.js, Anime.js, D3.js, Chart.js, Three.js, Voca.js, TensorFlow.js 등을 찾을 수 있습니다.

07-1 문서 객체 조작하기

1. ②

2. ① h1, #header, h1#header

② span, .active, span.active

③ input, input#name-input, input[type=text] 등

3. ④ 없는 속성입니다. textContent와 이름이 비슷해서 있을 수 있다고 생각할 수도 있으므로 주의하세요.

4. ①) borderRadius ② fontFamily ③ lineHeight ④ width ⑤ boxSizing

07-2 이벤트 활용

1. ⓐ-③, ⓑ-②, ⓒ-① **2.** ③

3. ⓐ-②2, ⓑ-①, ⓒ-③, ⓓ-④ **4.** ①

5. ②, ③

6. 다양하게 생각해볼 수 있습니다. 가장 쉽게는 02-3절의 6번에서 보았던 prompt() 함수와 alert() 함수를 사용해서 생각했던 입출력 예제들을 모두 입력 양식으로 옮길 수 있습니다. 또한 인터넷에서 자주 볼 수 있는 이메일 형식 확인, 주문 확인 등도 모두 만들어볼 수 있습니다.

08-1 구문 오류와 예외

1. ① 구문 오류가 발생합니다. 나머지는 모두 예외가 발생합니다.

2. ③ try - catch - finally 순서로 되어야 합니다. catch와 finally 둘 중 하나는 생략될 수 있습니다. 직접 코드를 입력하면서 확인해보세요.

3. ③ 구문 오류는 try catch finally 구문으로 처리할 수 없습니다.

08-2 예외 처리 고급

1. ④ **2.** ①

3. 콜백 함수를 함께 활용하면 그때부터 실행 흐름이 약간 이상해집니다. 현재 예제는 간단하지만, 이후에 Node.js 등을 공부할 때는 더 복잡해집니다. 어떤 구문을 보았을 때 예외 처리를 어떻게 할 수 있는 것인지 계속해서 생각해보면 좋습니다.

```
try 구문입니다
catch 구문입니다
finally 구문입니다
```

09-1 클래스의 기본 기능

1. ② C++, C#, 자바 등의 프로그래밍 언어에 해당합니다.

2~3. 문제에 나와있는 예를 기반으로 자신의 생각을 정리해보세요.

09-2 클래스의 고급 기능

1. ② extends로 s가 붙어야 합니다. ①과 혼동하면 안됩니다. 혼동된다면 코드를 입력할 때 색상이 변경되는지로 확인하세요.

2. ② **3.** ① **4.** ④

5. ④ **6.**

```
Parent.test() 메소드
ChildA.test() 메소드
ChildB.test() 메소드
Parent.test() 메소드
```

10-1 리액트의 기본

1. ③ **2.** ② 대소문자 등이 달라집니다. **3.** ①

4.

```
<script type="text/babel">
  // 애플리케이션 클래스 생성하기
  class App extends React.Component {
    constructor (props) {
      super(props)
      this.state = { seconds: 0 }
      this.handleChange = this.handleChange.bind(this)
    }

    handleChange (event) {
      if (event.target.checked) {
        this.timerId = setInterval(() => {
          this.setState({
            seconds: this.state.seconds + 1
```

```
          })
        }, 1000)
      } else {
        clearInterval(this.timerId)
      }
    }

    render () {
      return <div>
        <input type="checkbox" onChange={this.handleChange} />
        <span>타이머 활성화</span>
        <h1>{this.state.seconds}초</h1>
      </div>
    }
  }

  // 출력하기
  const container = document.getElementById('root')
  ReactDOM.render(<App />, container)
</script>
```

5. 코드를 분석해보면서 차근차근 이해해보세요. select에 selected 속성을 주어도 문제 없지만, 그렇게 하면 콘솔에 경고가 출력됩니다. "React를 사용할 때는 selected 대신 defaultValue를 사용해달라"라고 하기에 defaultValue라는 속성을 입력했습니다. value를 단순하게 value="10"이 아니라, value={10}으로 입력한 것은 숫자를 그대로 활용하기 위해서입니다.

```
<script type="text/babel">
  // 애플리케이션 클래스 생성하기
  class App extends React.Component {
    constructor (props) {
      super(props)
      this.state = {
        input: 0,
        output: 0,
        scale: 10
      }
```

```
      this.handleInput = this.handleInput.bind(this)
      this.handleSelect = this.handleSelect.bind(this)
    }

    handleInput (event) {
      const output = (event.target.value * this.state.scale).toFixed(2)
      this.setState({
        input: event.target.value,
        output: output
      })
    }
    handleSelect (event) {
      const output = (this.state.input * event.target.value).toFixed(2)
      this.setState({
        scale: event.target.value,
        output: output
      })
    }

    render () {
      return <div>
        <input
          value={this.state.input}
          onChange={this.handleInput}/>
        <span>cm = {this.state.output}</span>
        <select value={this.state.value} onChange={this.handleSelect}>
          <option defaultValue value={10}>mm</option>
          <option value={0.01}>m</option>
          <option value={0.393701}>inch</option>
        </select>
      </div>
    }
  }

  // 출력하기
  const container = document.getElementById('root')
  ReactDOM.render(<App />, container)
</script>
```

6.

```
<script type="text/babel">
  // 애플리케이션 클래스 생성하기
  class App extends React.Component {
    constructor (props) {
      super(props)
      this.state = {
        text: '',
        length: 0
      }
      this.handleChange = this.handleChange.bind(this)
    }

    handleChange (event) {
      this.setState({ text: event.target.value })
    }

    componentDidMount () {
      // 컴포넌트가 화면에 출력되었을 때
      this.timerId = setInterval(() => {
        this.setState({
          length: this.state.text.length
        })
      }, 500)
    }
    componentWillUnmount () {
      // 컴포넌트가 화면에서 제거될 때
      clearInterval(this.timerId)
    }

    render () {
      return <div>
        <h1>글자 수: {this.state.length}</h1>
        <textarea
          value={this.state.text}
          onChange={this.handleChange}></textarea>
      </div>
```

```
    }
  }

  // 출력하기
  const container = document.getElementById('root')
  ReactDOM.render(<App />, container)
</script>
```