CNU 국립 충남대학교 CHUNGNAM NATIONAL UNIVERSITY

26. 브라우저 어떻게 동작하는가?

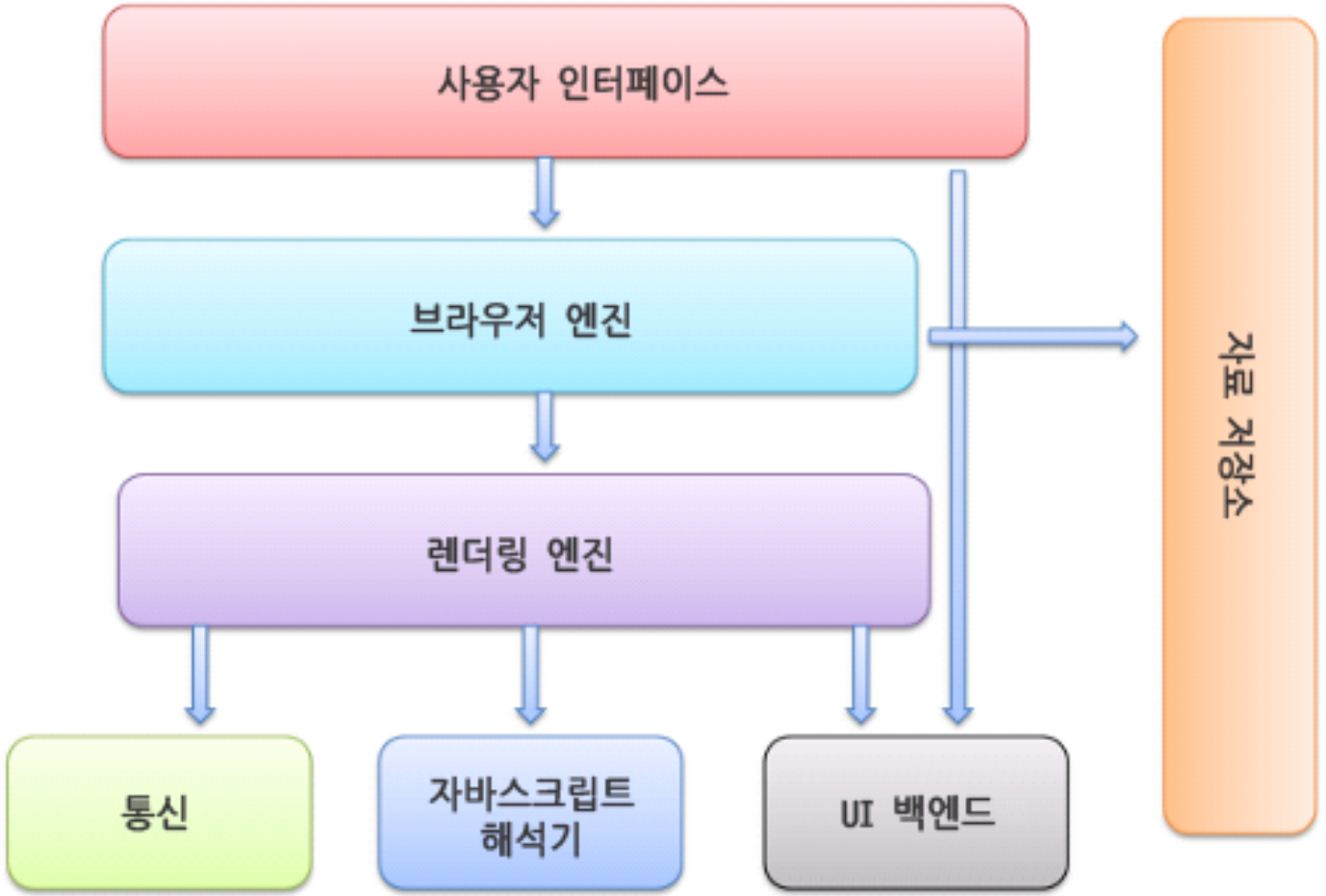
웹프로그래밍

2020년 2학기

충남대학교 컴퓨터공학과

"본 서비스는 교수/학생이 원격수업 목적으로 이용하고 있는 서비스 입니다."

브라우저의 기본 구조



브라우저의 기본 구조 (계속)

- **사용자 인터페이스**
  - 주소 표시줄, 이전/다음 버튼, 북마크 메뉴 등. 요청한 페이지를 보여주는 창을 제외한 나머지 모든 부분
- **브라우저 엔진**
  - 사용자 인터페이스와 렌더링 엔진 사이의 동작을 제어
- **렌더링 엔진**
  - 요청한 콘텐츠를 표시. 예를 들어 HTML을 요청하면 HTML과 CSS를 파싱하여 화면에 표시함

브라우저의 기본 구조 (계속)

- **통신**
  - HTTP 요청과 같은 네트워크 호출에 사용됨. 이것은 플랫폼 독립적인 인터페이스이고 각 플랫폼 하부에서 실행됨
- **UI 백엔드**
  - 콤보 박스와 창 같은 기본적인 그림. 플랫폼에서 명시하지 않은 일반적인 인터페이스로서, OS 사용자 인터페이스 체계를 사용
- **자바스크립트 해석기**
  - 자바스크립트 코드를 해석하고 실행

브라우저의 기본 구조 (계속)

- 자료 저장소
  - 이 부분은 자료를 저장하는 계층이다. 쿠키를 저장하는 것과 같이 모든 종류의 자원을 하드 디스크에 저장할 필요가 있다. HTML5 명세에는 브라우저가 지원하는 ' Web storage'가 정의되어 있다

렌더링 엔진

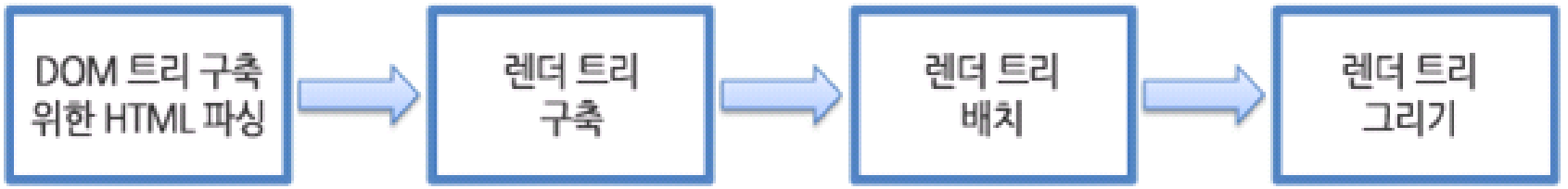
- 역할
  - 요청 받은 내용을 브라우저 화면에 표시하는 일
  - HTML 및 XML 문서와 이미지를 표시할 수 있다. 물론 플러그인이나 브라우저 확장 기능을 이용해 PDF와 같은 다른 유형도 표시할 수 있음

렌더링 엔진들

- **Internet Explorer (Microsoft)**
  - Rendering Engine : Trident
- **Chrome (Google)**
  - Rendering Engine : 27버전 이하 WebKit / 28버전 이상 Blink
- **Firefox (Mozilla)**
  - Rendering Engine : Gecko
- **Safari (Apple)**
  - Rendering Engine : WebKit
- **Opera (Opera)**
  - Rendering Engine : 14버전 이하 Presto / 15버전 이상 Blink

동작 과정

기본적인 동작 과정



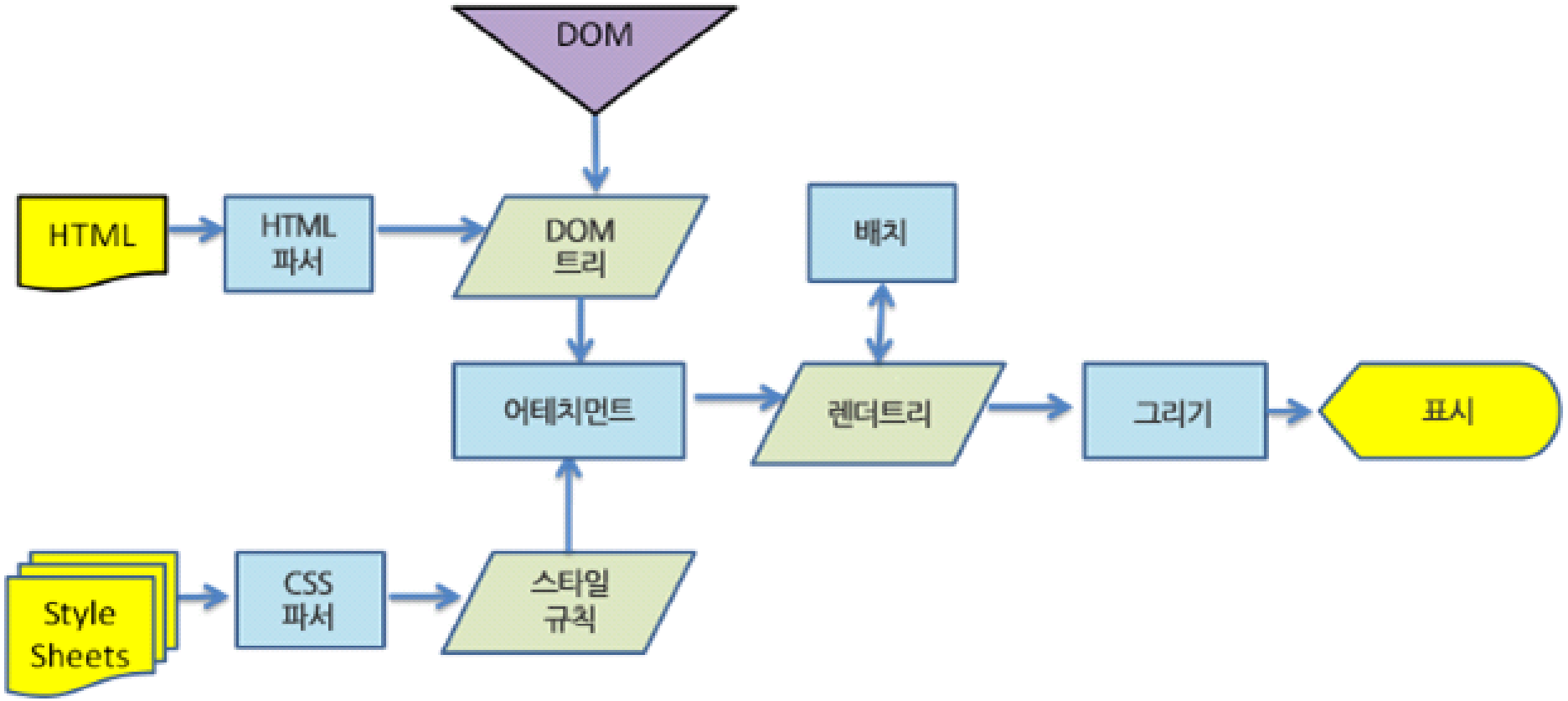
- 렌더링 엔진은 HTML 문서를 파싱하고 "콘텐츠 트리" 내부에서 태그를 DOM 노드로 변환.그 다음 외부 CSS 파일과 함께 포함된 스타일 요소도 파싱. 스타일 정보와 HTML 표시 규칙은 "렌더 트리"라고 부르는 또 다른 트리를 생성
- 렌더 트리는 색상 또는 면적과 같은 시각적 속성이 있는 사각형을 포함하고 있는데 정해진 순서대로 화면에 표시

동작 과정 (계속)

- 렌더 트리 생성이 끝나면 배치가 시작되는데 이것은 각 노드가 화면의 정확한 위치에 표시되는 것을 의미
- 다음은 UI 백엔드에서 렌더 트리의 각 노드를 방문하며 형상을 만들어 내는 그리기 과정임

동작 과정 예

- 웹킷 동작 과정



객체 모델 생성

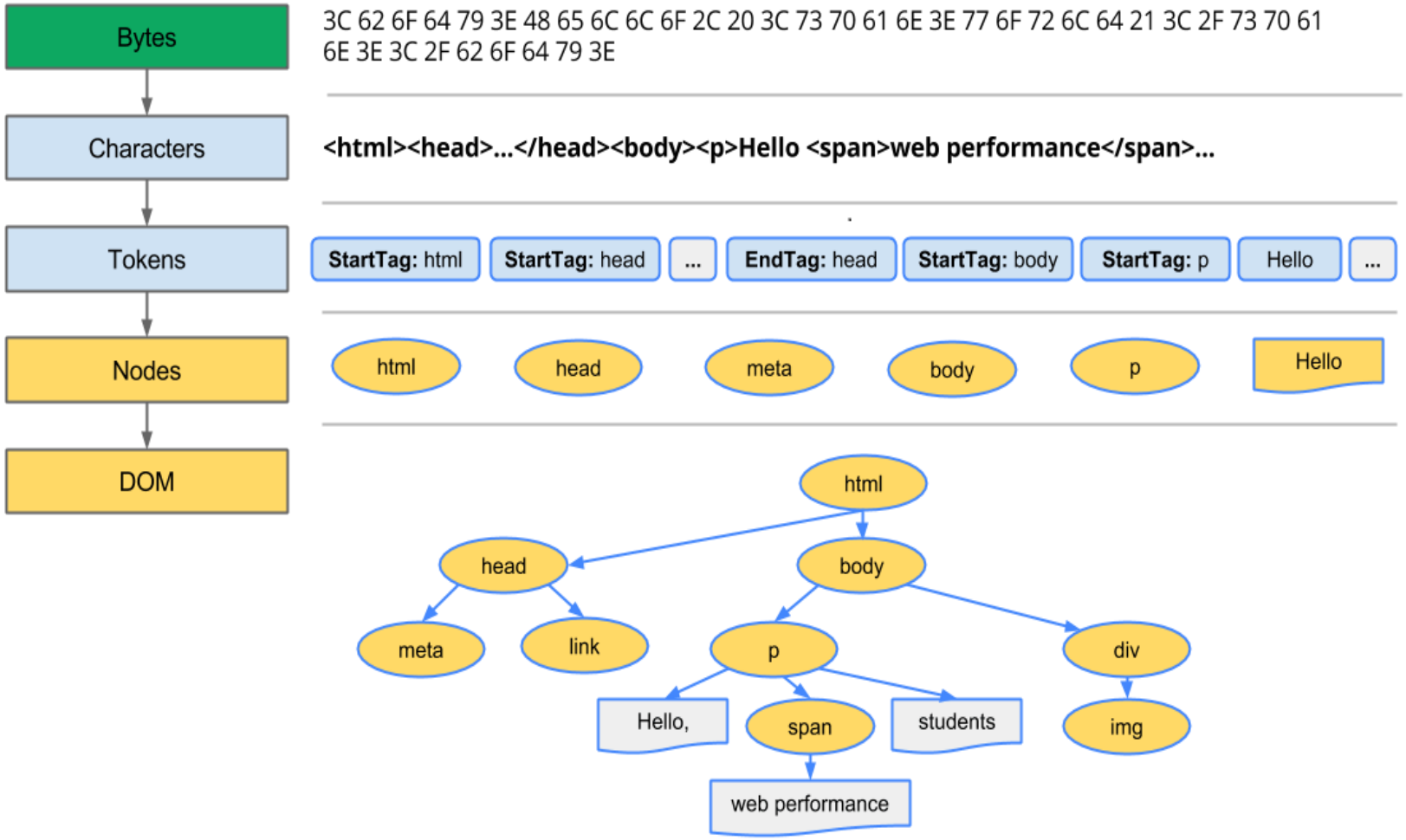
- 바이트 → 문자 → 토큰 → 노드 → 객체 모델
- HTML 마크업은 DOM(Document Object Model)으로 변환되고, CSS 마크업은 CSSOM(CSS Object Model)으로 변환
- DOM 및 CSSOM은 서로 독립적인 데이터 구조

객체 모델 생성 (계속)

- **DOM (Document Object Model) 으로의 변환 예**
  - 예) HTML 화일

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <link href="style.css" rel="stylesheet">
    <title>Critical Path</title>
  </head>
  <body>
    <p>Hello <span>web performance</span> students!</p>
    <div><img src="awesome-photo.jpg"></div>
  </body>
</html>
```

객체 모델 생성 (계속)


Bytes
Characters
Tokens
Nodes
DOM
3C 62 6F 64 79 3E 48 65 6C 6C 6F 2C 20 3C 73 70 61 6E 3E 77 6F 72 6C 64 21 3C 2F 73 70 61 6E 3E 3C 2F 62 6F 64 79 3E
<html><head>...</head><body><p>Hello web performance...
StartTag: html
StartTag: head
...
EndTag: head
StartTag: body
StartTag: p
Hello
...
html
head
meta
body
p
Hello
html
head
body
meta
link
p
div
Hello,
span
students
img
web performance

객체 모델 생성 (계속)

- 변환
  - 브라우저가 HTML의 원시 바이트를 디스크나 네트워크에서 읽어와서, 해당 파일에 대해 지정된 인코딩(예: UTF-8)에 따라 개별 문자로 변환
- 토큰화
  - 브라우저가 문자열을 W3C HTML5 표준에 지정된 고유 토큰으로 변환
    - 예) '<html>', '<body>' 및 꺽쇠 괄호로 묶인 기타 문자열
  - 각 토큰은 특별한 의미와 고유한 규칙을 가짐
- 렉싱
  - 방출된 토큰은 해당 속성 및 규칙을 정의하는 '객체'로 변환
- DOM 생성
  - 마지막으로, HTML 마크업이 여러 태그(일부 태그는 다른 태그 안에 포함되어 있음) 간의 관계를 정의하기 때문에 생성된 객체는 트리 데이터 구조 내에 연결
  - 이 트리 데이터 구조에는 원래 마크업에 정의된 상위-하위 관계도 포함
    - 예 : *HTML* 객체는 *body* 객체의 상위이고, *body* 는 *paragraph* 객체의 상위

객체 모델 생성 (계속)

- **CSSOM (CSS Object Model) 으로의 변환 예**
  - 예) CSS 파일

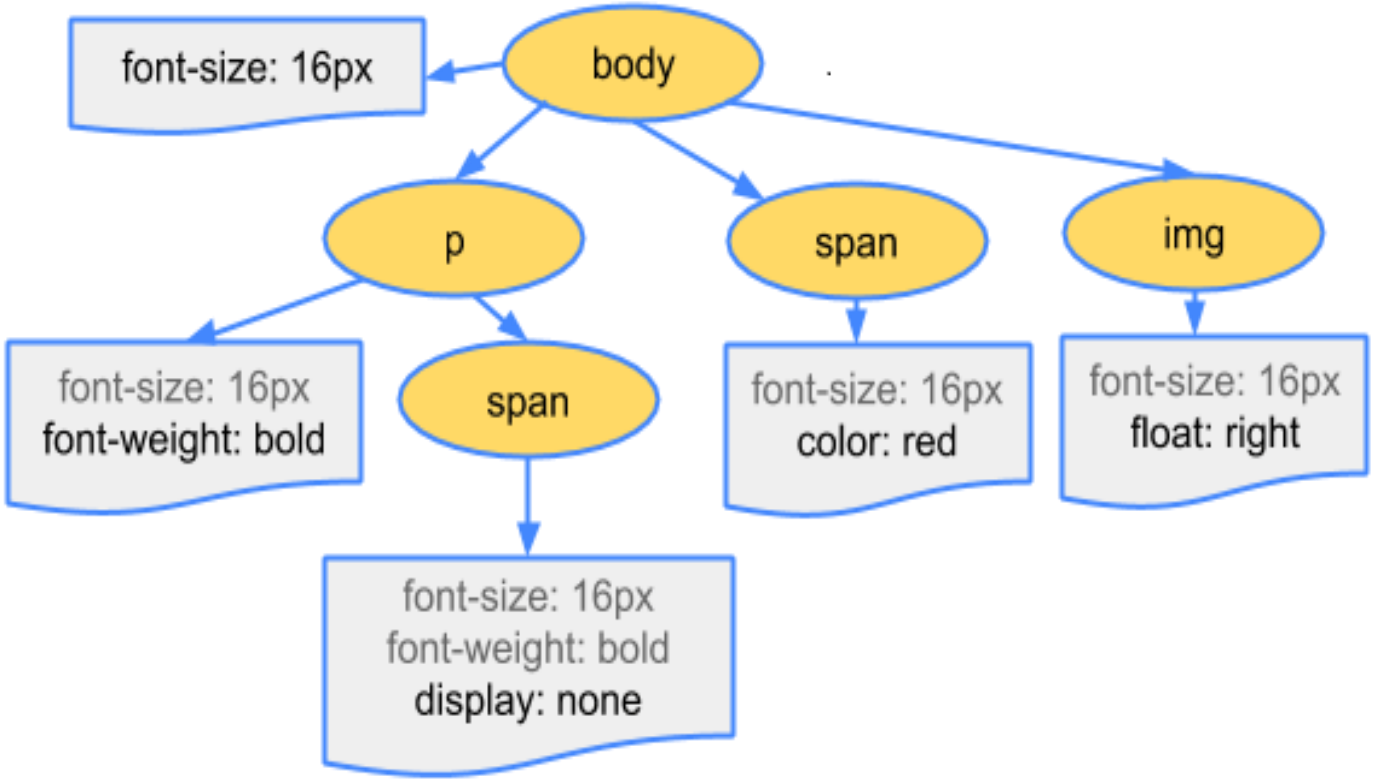
```
body { font-size: 16px }
p { font-weight: bold }
span { color: red }
p span { display: none }
img { float: right }
```

객체 모델 생성 (계속)

- HTML과 마찬가지로, 수신된 CSS 규칙을 브라우저가 이해하고 처리할 수 있는 형식으로 변환해야 함. 따라서 HTML 대신 CSS에 대해 HTML 프로세스를 반복

객체 모델 생성 (계속)

- CSS 바이트가 문자로 변환된 후 차례로 토큰과 노드로 변환되고 마지막으로 'CSS Object Model'(CSSOM)이라는 트리 구조에 링크

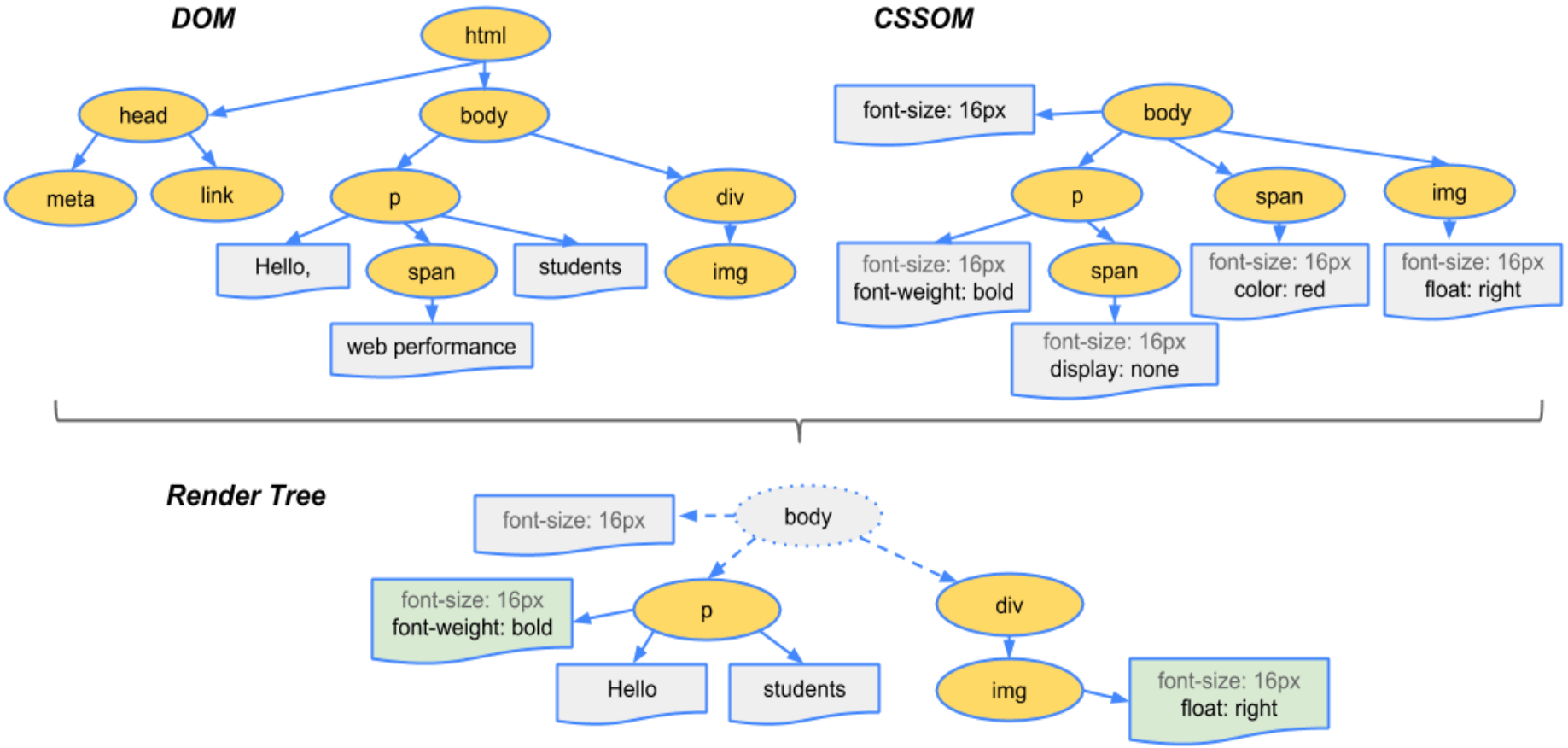


객체 모델 생성 (계속)

- CSSOM이 트리 구조를 가지는 이유
  - 페이지에 있는 객체의 최종 스타일을 계산할 때 브라우저는 해당 노드에 적용 가능한 가장 일반적인 규칙(예: body 요소의 하위인 경우 모든 body 스타일 적용)으로 시작한 후 더욱 구체적인 규칙을 적용하는 방식으로, 즉 '하향식'으로 규칙을 적용하는 방식으로 계산된 스타일을 재귀적으로 세분화

렌더링 트리 생성

- CSSOM 및 DOM 트리는 결합하여 렌더링 트리를 형성



렌더링 트리 생성 (계속)

- **렌더링 트리 생성을 위한 브라우저 동작**
  - DOM 트리의 루트에서 시작하여 표시되는 노드 각각을 Traverse함
    - 일부 노드는 표시되지 않으며(예: 스크립트 태그, 메타 태그 등), 렌더링된 출력에 반영되지 않으므로 생략
    - 일부 노드는 CSS를 통해 숨겨지며 렌더링 트리에서도 생략됨.
      - 예 : span 노드의 경우 'display: none' 속성을 설정하는 명시적 규칙이 있기 때문에 렌더링 트리에서 누락
  - 표시된 각 노드에 대해 적절하게 일치하는 CSSOM 규칙을 찾아 적용
  - 표시된 노드를 콘텐츠 및 계산된 스타일과 함께 내보냄

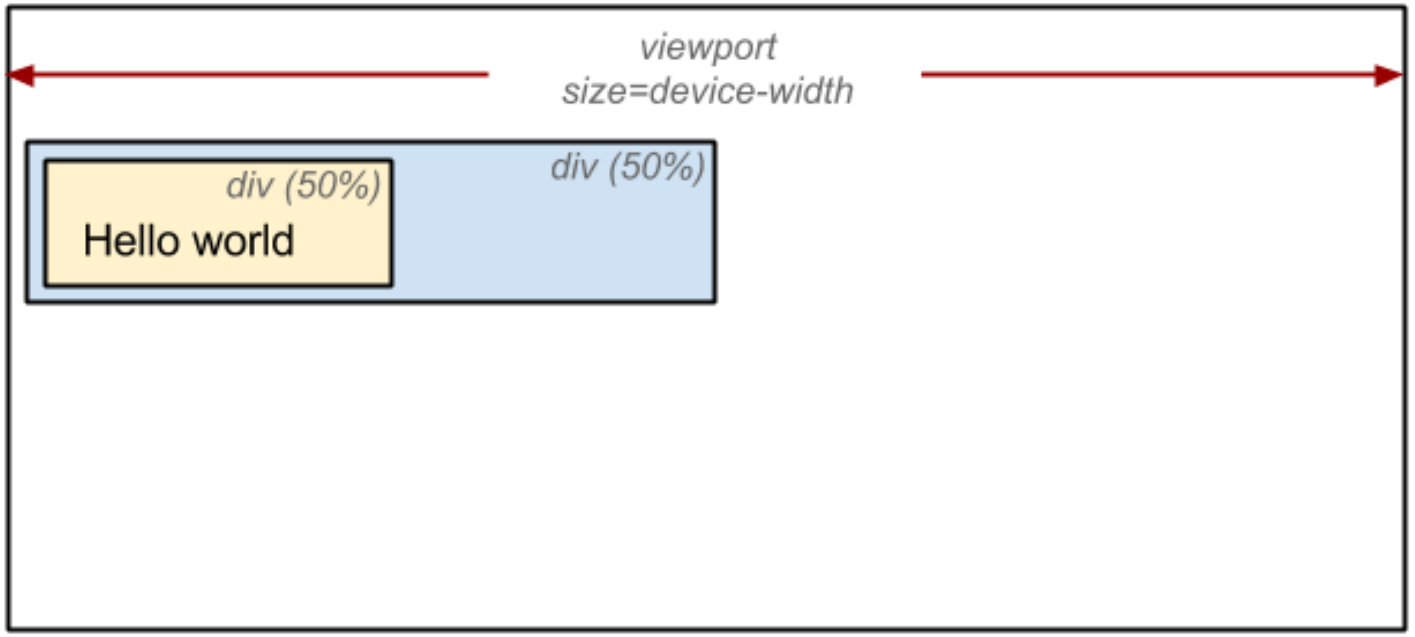
레이 아웃 및 페인트

- 지금까지 표시할 노드와 해당 노드의 계산된 스타일을 계산했음. 하지만 기기의 뷰 포트 내에서 이러한 노드의 정확한 위치와 크기를 계산하지는 않았음. 이것이 바로 '레이아웃' 단계이며, 경우에 따라 '리플로우'라고도 함.
- 예) 레이 아웃 설명을 위한 예

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>Critial Path: Hello world!</title>
  </head>
  <body>
    <div style="width: 50%">
      <div style="width: 50%">Hello world!</div>
    </div>
  </body>
</html>
```

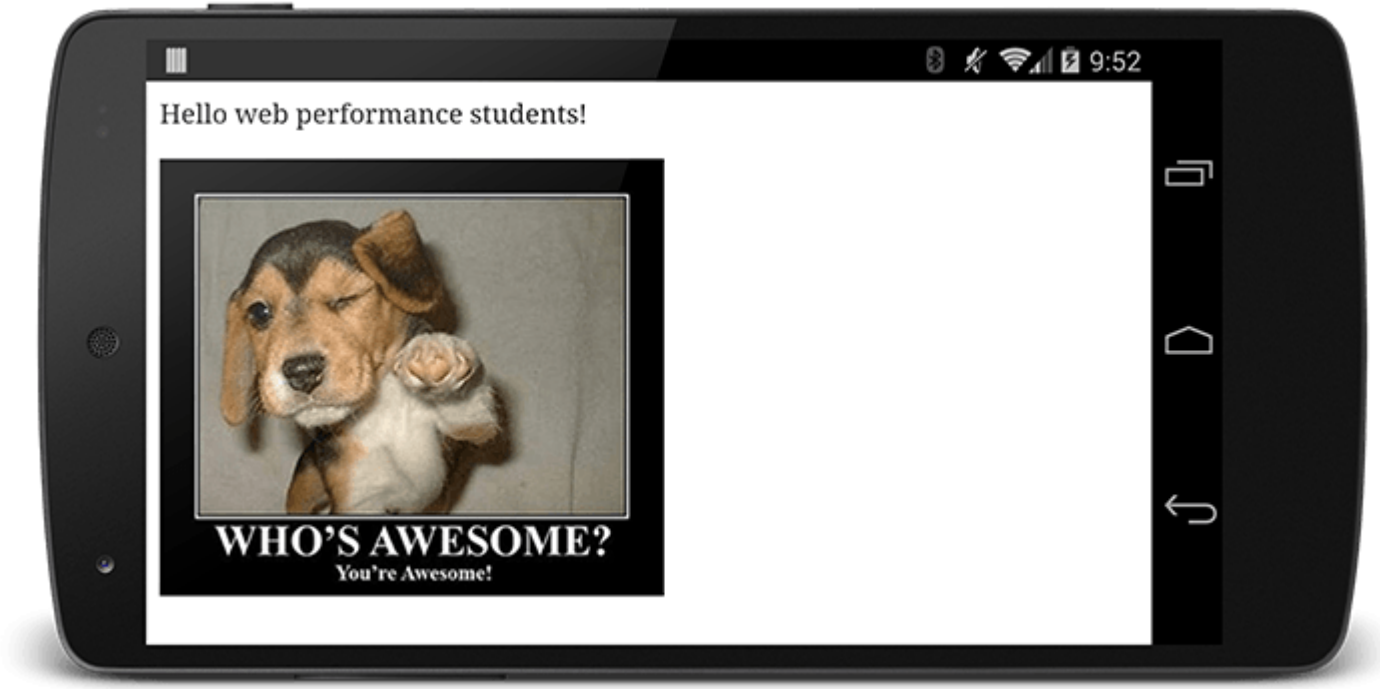
레이 아웃 및 페인트 (계속)

- 레이아웃 프로세스에서는 뷰포트 내에서 각 요소의 정확한 위치와 크기를 정확하게 캡처하는 '상자 모델'이 출력됨.
- 모든 상대적인 측정값은 화면에서 절대적인 픽셀로 변환됨.



레이 아웃 및 페인트 (계속)

- 이제 표시되는 노드와 해당 노드의 계산된 스타일 및 기하학적 형태에 대해 파악했으므로, 렌더링 트리의 각 노드를 화면의 실제 픽셀로 변환하는 마지막 단계로 이러한 정보를 전달할 수 있음. 이 단계를 흔히 '페인팅' 또는 '래스터화'라고 함



레이 아웃 및 페인트 (계속)

- 렌더링 트리 생성, 레이아웃 및 페인트 작업을 수행하는 데 필요한 시간은 문서의 크기, 적용된 스타일 및 실행 중인 기기에 따라 달라짐.

- 즉, 문서가 클수록 브라우저가 수행해야 하는 작업도 더 많아지며, 스타일이 복잡할수록 페인팅에 걸리는 시간도 늘어남.
  - 예 : 단색은 페인트하는 데 시간과 작업이 적게 필요
  - 예 : 그림자 효과는 계산하고 렌더링하는 데 시간과 작업이 더 필요

자바스크립트로 상호 작용 추가

- 문서에서 스크립트의 위치는 중요
- 브라우저가 스크립트 태그를 만나면 이 스크립트가 실행 종료될 때까지 DOM 생성이 일시 중지
- 자바스크립트는 DOM 및 CSSOM을 Query하고 수정할 수 있음
- 자바스크립트 실행은 CSSOM이 준비될 때까지 일시 중지

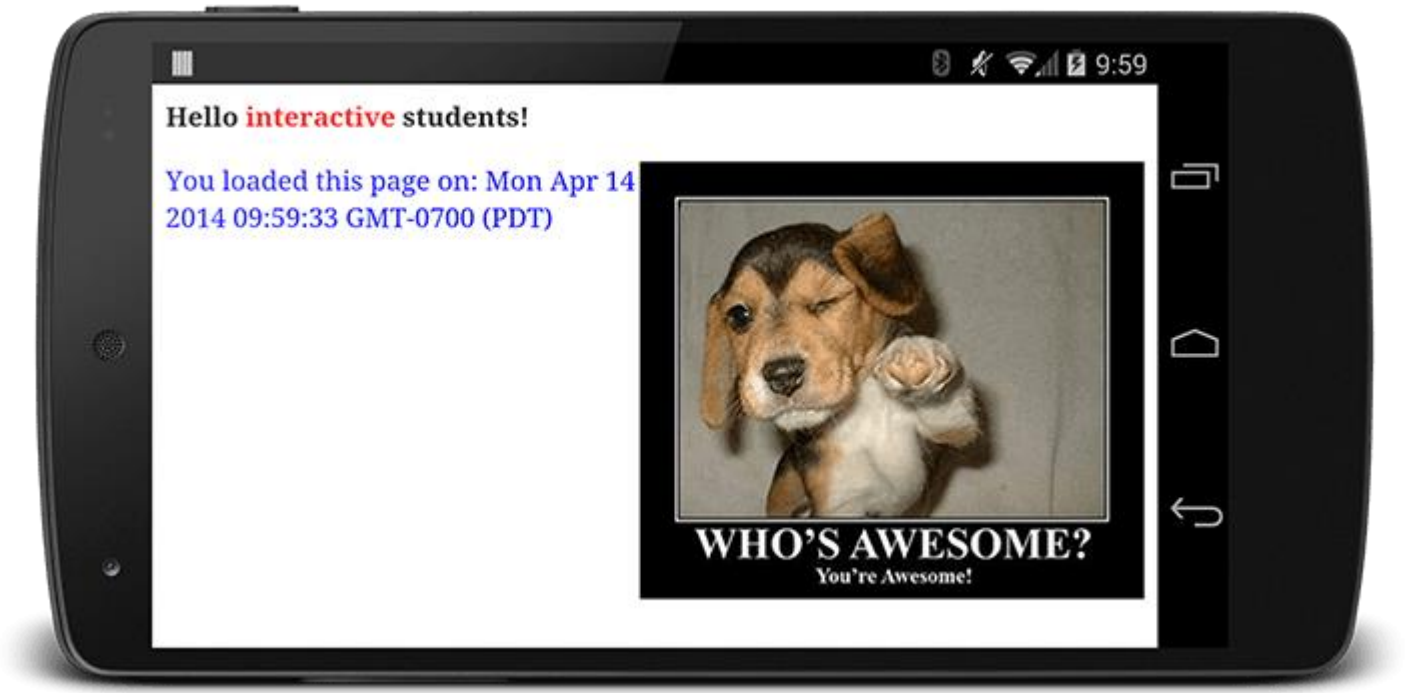
자바스크립트로 상호 작용 추가 (계속)

- 예) 자바스크립트와의 상호 작용 설명을 위한 예

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <link href="style.css" rel="stylesheet">
    <title>Critical Path: Script</title>
  </head>
  <body>
    <p>Hello <span>web performance</span> students!</p>
    <div><img src="awesome-photo.jpg"></div>
    <script>
      var span = document.getElementsByTagName('span')[0];
      span.textContent = 'interactive'; // change DOM text content
      span.style.display = 'inline';  // change CSSOM property
      // create a new element, style it, and append it to the DOM
      var loadTime = document.createElement('div');
      loadTime.textContent = 'You loaded this page on: ' + new Date();
      loadTime.style.color = 'blue';
      document.body.appendChild(loadTime);
    </script>
  </body>
</html>
```

자바스크립트로 상호 작용 추가 (계속)

- 26 페이지 예의 결과 화면



참고 자료

- How Browsers Work : Behind the scenes of modern web browsers, https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

- 브라우저는 어떻게 동작하는가?, 2012, 5, 17, https://d2.naver.com/helloworld/59361

- 객체 모델 생성,Google Developer Site, https://developers.google.com/web/fundamentals/performance/critical-rendering-path/constructing-the-object-model