"소극적인 노력은 고생이다. 적극적인 노력은 훈련이다."

Lecture 2
Introducing JavaScript and the DOM: A Little Code

Samkeun Kim skim@hknu.ac.kr

http://cyber.hknu.ac.kr/

국립한경대학교
HANKYONG NATIONAL UNIV.

JavaScript 동작원리

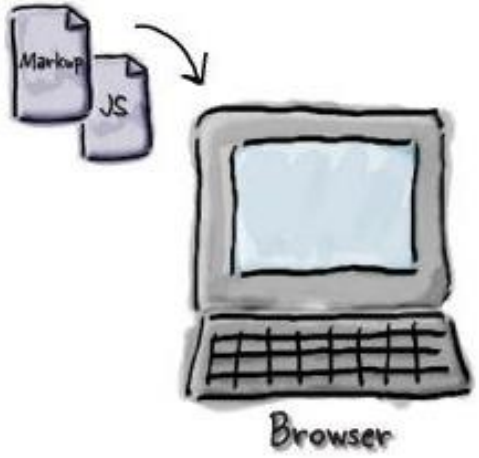
Our goal is to write JavaScript code that runs in the browser when your web page is loaded

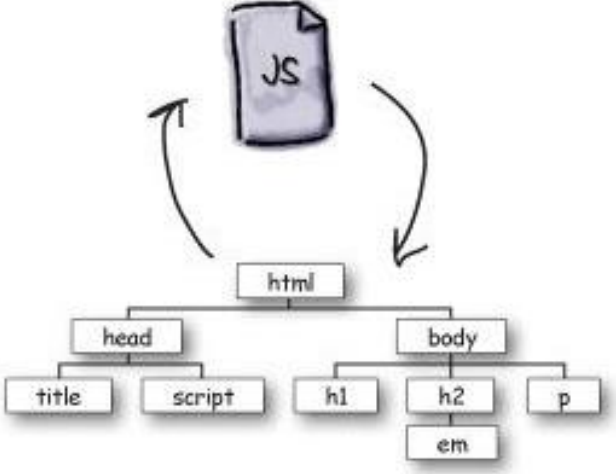
That code might

- **respond** to user actions,
- **update** or **change** the page,
- communicate with **web services (web API)**, and
- make your page feel more like an **application** than a document.

Let's look at how all that works:

```
<html>
<head>
<script>
 var x = 49;
</script>
<body>
<h1>My first JavaScript</h1>
<p></p>
<script>
 x = x + 2;
</script>
</body>
</html>
```





Writing

1

You create your HTML markup and your JavaScript code and put them in files, say index.html and index.js (or they both can go in the HTML file).

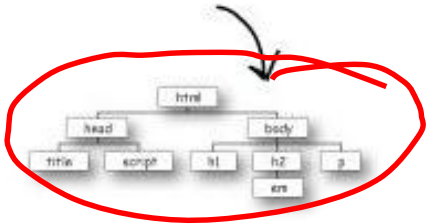
Loading

2

The browser retrieves and loads your page, parsing its contents from top to bottom.

As it encounters JavaScript, the browser parses the code and checks it for correctness, and then executes the code.

The browser also builds an internal model of the HTML page, called the DOM.



Running

3

JavaScript continues executing, using the DOM to examine the page, change it, receive events from it, or ask the browser to retrieve other data from the web server.

JavaScript로 무엇을 할 수 있을까?

Once you've got a page with a **<script>** element (or a reference to a separate JavaScript file), you're ready to start coding.

JavaScript is a **full-fledged** programming language and you can do pretty much anything with it you can with other languages, and even more because we're programming inside a web page!

You can tell JavaScript to:

1. Make a statement

Create a variable and assign values,

add things together, calculate things,

use built-in functionality from a JavaScript library.

```
var temp = 98.6;
var beanCounter = 4;
var reallyCool = true;
var motto = "I Rule";
temp = (temp - 32) * 5 / 9;
motto = motto + " and so do you!";
var pos = Math.random();
```

2. Do things more than once, or twice

Perform statements over and over, as many times as you need to.

```
while (beanCounter > 0) {
   processBeans();
   beanCounter = beanCounter - 1;
}
```

3. Make decisions

Write code that is conditional, depending on the state of your app.

```
if (isReallyCool) {
   invite = "You're invited!";
} else {
  invite = "Sorry, we're at capacity.";
}
```

변수 선언

Variables hold things.

With JavaScript they can hold lots of different things.

Let's declare a few variables that hold things:

```
var winners = 2;
var boilingPt = 212.0;
var name = "Dr. Evil";
var isEligible = false;
```

Integer numeric values.

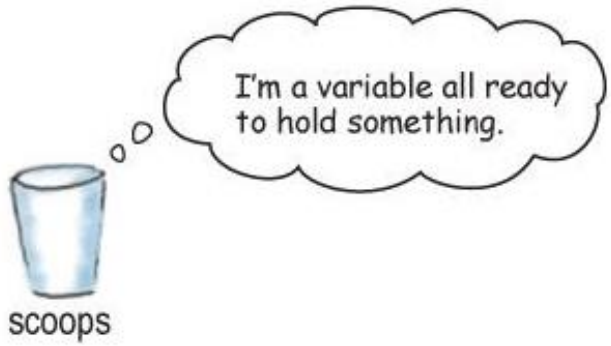
Or floating point numeric values.

Or, strings of characters (we call those "strings," for short).

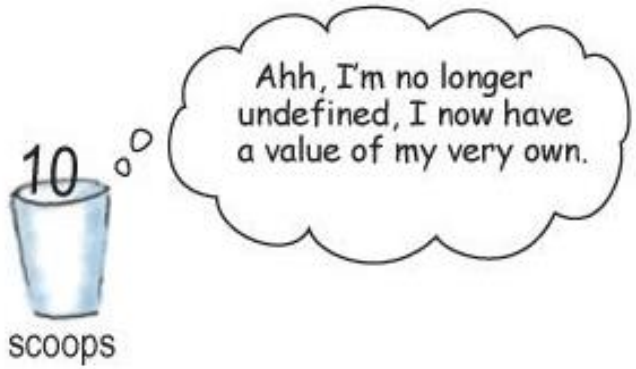
Or a boolean value, which is true or false.

Three steps of creating a variable

1. 변수 scoops를 선언한다



3. 값을 변수에 할당한다



2. 변수에 지정할 값이 필요하다

Your value can be a literal value, like a number or a string.

```
var scoops = 10;
var scoops = totalScoops / people;
var scoops = Math.random() * 10;
```

Or, the value can be the result of an expression.

Or use one of JavaScript's internal library functions, like a random number generator, to create a value. More on this and your own functions later.

How to name your variables – Rule #1

Rule #1: Start your variables with a letter, an underscore or a dollar sign.

Here are some examples:

```
var thisIsNotAJoke;
var _myVariable;
var $importantVar;
```

Do this...

Begins with number, not good. → `var 3zip;`

Begin with symbols (% and ~) that aren't allowed. → `var %entage;` → `var ~approx;`

...not this.

How to name your variables – Rule #2

Rule #2:
Then you can use **any number** of letters, numeric digits, underscores or dollar signs.

```
var my3sons;
var cost$;
var vitaminB12;
```

Do this...

Got a space, not allowed

```
var zip code;
```

Got –, + signs. Not allowed and will seriously confuse JavaScript.

```
var first-name;
var to+do;
```

...not this.

How to name your variables – Rule #3

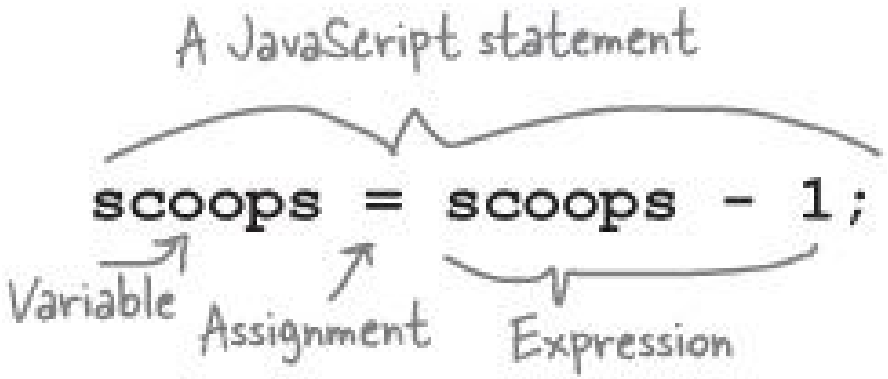
Rule #3:
Make sure you avoid all of JavaScript's **reserved** words

abstract	delete	goto	null	throws
as	do	if	package	transient
boolean	double	implements	private	true
break	else	import	protected	try
byte	enum	in	public	typeof
case	export	instanceof	return	use
catch	extends	int	short	var
char	false	interface	static	void
class	final	is	super	volatile
continue	finally	long	switch	while
const	float	namespace	synchronized	with
debugger	for	native	this	
default	function	new	throw	

NOTE
Avoid these as variable names!

Getting Expressive

We've already seen some JavaScript statements that look like:



Getting Expressive – Closer look

You can write expressions that result in numbers...

Numeric expressions

```
(9 / 5) * tempC + 32
x - 1
Math.random() * 10
2.123 + 3.2
```

You can write expressions that result in the boolean values true or false (these are, obviously, boolean expressions).

Boolean expressions

```
2 > 3
startTime > now
tempF < 75
level == 4
pet == "Duck"
```

...and you can write expressions that result in strings.

String expressions

```
"super" + "cali" + youKnowTheRest
P.innerHTML
"March" + "21" + "st"
phoneNumber.substring(0, 3)
```

There are other types of expressions too; we'll be getting to these later.

Other expressions

```
function () {...}
document.getElementById("pink")
new Array(10)
```

실습과제 2-1 Getting Expressive

```
(9 / 5) * tempC + 32
```

What's the result when tempC is 10? _____

```
"Number" + " " + "2"
```

What's the resulting string? _____

```
level >= 5
```

What's the result when level is 10? _____

How about when level is 5? _____

```
color != "pink"
```

Hint: ! means not.

What's the result if color is "blue"? _____

```
(2 * Math.PI) * r
```

What's the result if r is 3? _____

Hint: Math.PI gives you the value of pi (you know, 3.14....)

Not this kind of expression!

실습과제 2-2 Getting Expressive

Circle the statements that are legal

```
var x = 1138;
var y = 3/8;
var s = "3-8";
x = y;
var n = 3 - "one";
var t = "one" + "two";
var 3po = true;
var level_ = 11;
var highNoon = false;
var $ = 21.30;
var z = 2000;
var isBig = y > z;
z = z + 1;
z--;
z y;
x = z * t;
while (highNoon) {
    z--;
}
```

Getting Expressive – Converting

message = 2 + "if by sea";

- we know that + could be for adding numbers together, and it's also the operator used to concatenate strings together.

value = 2 * 3.1;

- JavaScript **converts** the integer 2 into a **floating point** number and the result is 6.2.

---------- BRAIN POWER ----------

- What does JavaScript evaluate the following statements to?

numORString1 = "3" + "4"

numORString2 = "3" * "4"

And why?

Doing things over and over... - while

JavaScript – **while loop**: 조건이 만족될 때까지 뭔가를 수행한다.

조건 테스트

We've got a tub of ice cream, and it's got ten scoops left in it. Here's a variable declared and initalized to ten.

```
var scoops = 10;

while (scoops > 0) {
   alert("More icecream!");
   scoops = scoops - 1;
}

alert("life without ice cream isn't the same");
```

While uses a boolean expression that evaluates to true or false. If true, the code after it is executed.

While there are more than zero scoops left, we're going to keep doing everything in this code block.

Each time through the while loop, we alert the user there is more ice cream, and then we take one scoop away by subtracting one from the number of scoops.

When the condition (scoops > 0) is false, the loop is done, and the code execution continues here, with whatever the next line of your program is.

Doing things over and over... - while

1. 값을 초기화한다.
2. while 루프를 테스트한다.
3. True 이면 코드 블록을 실행한다.
4. 어떤 시점에서 조건이 실패하여 루프가 종료되도록 조건 테스트와 관련된 값을 업데이트한다.

```
var scoops = 10;

while (scoops > 0) {
    alert("More icecream!");
    scoops = scoops - 1;
}

alert("life without ice cream isn't the same");
```

INITIALIZE

DO CONDITIONAL TEST

EXECUTE CODE BLOCK WHILE CONDITIONAL TEST IS TRUE

UPDATE

CONTINUE AFTER LOOP CONDITION FAILS

Doing things over and over... - for

JavaScript – **for 루프 제공**

Ice cream 코드:

INITIALIZE

DO CONDITIONAL TEST

UPDATE

```
for (scoops = 10; scoops > 0; scoops--) {
   alert("There's more ice cream!");
}
alert("life without ice cream isn't the same");
```

EXECUTE CODE BLOCK WHILE CONDITIONAL TEST IS TRUE

CONTINUE AFTER LOOP CONDITION FAILS

실습과제 2-3 Be the Browser

Snippet 1

```
var count = 0;
for (var i = 0; i < 5; i++) {
    count = count + i;
}
alert("count is " + count);
```

What count does the alert show?

↳ _____

Snippet 2

```
var tops = 5;
while (tops > 0) {
    for (var spins = 0; spins < 3; spins++) {
        alert("Top is spinning!");
    }
    tops = tops - 1;
}
```

_____

How many times do you see the alert, "Top is spinning!"?

실습과제 2-3 Be the Browser

Snippet 3

```
for (var berries = 5; berries > 0; berries--) {
    alert("Eating a berry");
}
```

How many berries did you eat? → _____

Snippet 4

```
for (scoops = 0; scoops < 10; scoop++) {
   alert("There's more ice cream!");
}
alert("life without ice cream isn't the same");
```

_____ How many scoops of ice cream did you eat?

```
if (cashInWallet > 5) {
  order = "I'll take the works: cheeseburger, fries and a coke";
} else {
  order = "I'll just have a glass of water";
}
```

Make decisions with JavaScript

for / while 문에서: 루핑을 계속해야 하는지 여부를 결정하기 위해 조건 테스트에서 부울 식을 사용해 왔다.

<u>조건 문에서도 부울 식을 사용할 수 있다:</u>

Here's our boolean expression, testing to see how many scoops are left.

If there are < 3 scoops left we then execute the code block.

```
if (scoops < 3) {
    alert("Ice cream is running low!");
}
```

We can string together more than one test too:

Add as many tests with "else if" as you need, each with its own associated code block that will be executed when the condition is true.

```
if (scoops < 3) {
    alert("Ice cream is running low!");
} else if (scoops > 9) {
    alert("Eat faster, the ice cream is going to melt!");
}
```

Making more decisions... and, adding a catchall

Catchall 사용 - 모든 조건이 실패하면 실행되는 마지막 else 문

Notice we changed this to only happen when scoops is precisely 3.

```
if (scoops == 3) {
   alert("Ice cream is running low!");
} else if (scoops > 9) {
   alert("Eat faster, the ice cream is going to melt!");
} else if (scoops == 2) {
   alert("Going once!");
} else if (scoops == 1) {
   alert("Going twice!");
} else if (scoops == 0) {
   alert("Gone!");
} else {
   alert("Still lots of ice cream left, come and get it.");
}
```

We've added additional conditions to have a countdown to zero scoops.

Here's our catchall; if none of the conditions above are true, then this block is guaranteed to execute.

실습과제 2-4

Take the **code before** and insert it into the while loop below. Walk through the while loop and write down the alerts in the sequence they occur.

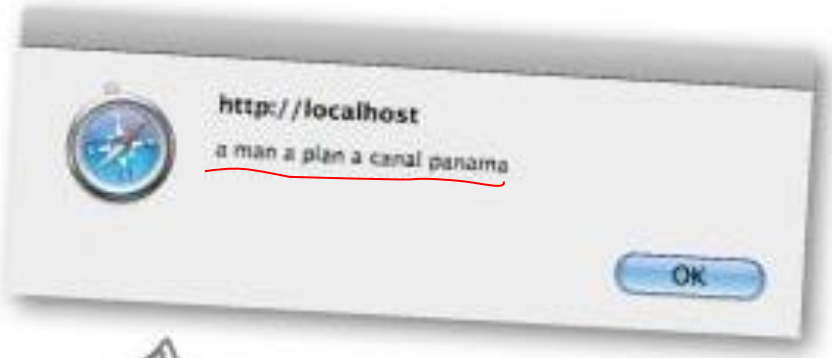
```
var scoops = 10;
while (scoops >= 0) {
                    } Insert the code above here...
   scoops = scoops - 1;
}
alert("Life without ice cream isn't the same");
```

실습과제 2-5 Code Magnets

```
var word1 = "a";
var word2 = "nam";
var word3 = "nal p";
var word4 = "lan a c";
var word5 = "a man a p";

var phrase = "";

for (var i = 0; ______; _____) {
    if (i == 0) {
        phrase = _______________;
    }
    else if (i == 1) {
        phrase = _______________ + word4;
    }
    ________ (i == 2) {
        _______________ = phrase + word1 + word3;
    }
    ________ (_______) {
        phrase = phrase + ________ + word2 + word1;
    }
}
alert(phrase);
```

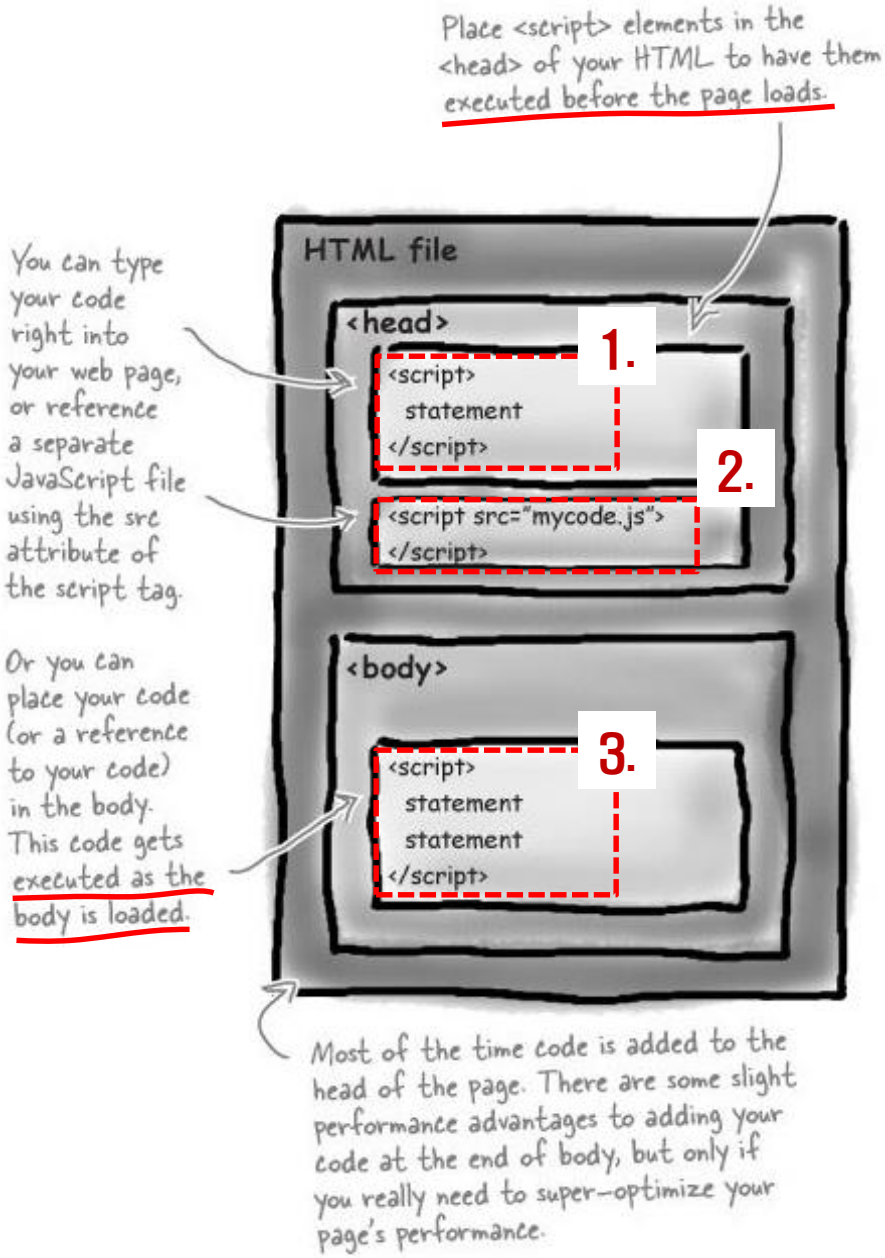


A palindrome is a sentence that can be read the same way backwards and forwards! Here's the palindrome you should see if the magnets are all in the right places.

```
else if (i == 0)
word5
i == 3
i++
else if
phrase
i == 4
word2
word4
i < 3
word1
i < 4
else
word0
i = 0
+
i--
i = 3
word3
```


I was told we'd be putting JavaScript in *our web pages*. When are we going to get there, or are we just going to keep playing around with JavaScript?

How and where to add JavaScript to your pages



JavaScript 코드를 웹 페이지에 추가하기 위한 세가지 방법:

1. Place your script inline, in the **<head>** element.

2. Add your script by referencing a **separate JavaScript file**.

3. Add your code in the **body** of the document, either inline or as a link to a separate file.

How JavaScript interacts with your page

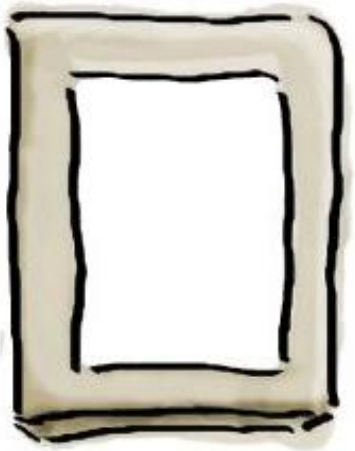
JavaScript and *HTML* are **two different things**

HTML is **markup** and JavaScript is **code**

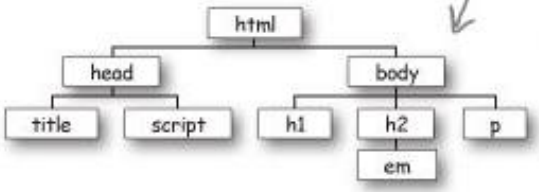
So how do you get JavaScript to interact with the markup in your page?

You use the **Document Object Model (DOM)**

Your browser



① 페이지가 브라우저에 로드되면 HTML을 파싱하고 HTML 마크업의 모든 엘리먼트를 포함하는 내부 문서 모델을 생성한다.



우리는 이것을 Document Object Model이라고 부른다.

DOM

간단하게 DOM이라고 부르기도 한다!!

② JavaScript는 DOM과 상호 대화하여 그들의 엘리먼트 및 내용에 접근할 수 있다.

③ JavaScript가 DOM을 수정하면 브라우저가 그 페이지를 동적으로 업데이트한다.

It's tl
to, and changing the DOM that JavaScript can be used to write interactive web pages/apps. This book will show you how.

How to bake your very own DOM

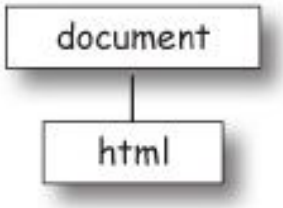
Ingredients - One well-formed HTML5 page

```
<!doctype html>
<html lang="en">
<head>
  <title>My blog</title>
  <meta charset="utf-8">
  <script src="blog.js"></script>
</head>
<body>
  <h1>My blog</h1>
  <div id="entry1">
     <h2>Great day bird watching</h2>
     <p>
       Today I saw three ducks!
       I named them
       Huey, Louie, and Dewey.
     </p>
     <p>
       I took a couple of photos...
     </p>
   </div>
 </body>
 </html>
```

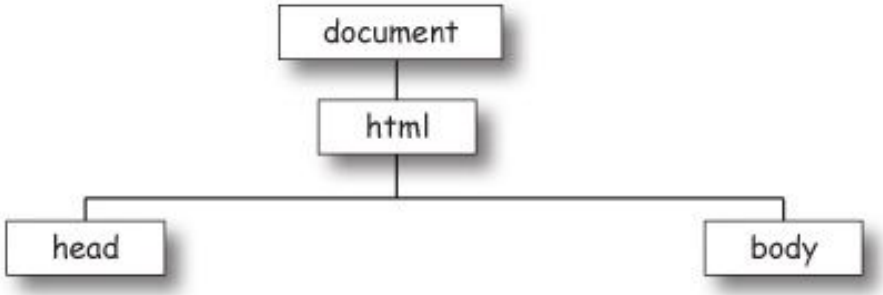
How to bake your very own DOM

1. Start by creating a document node at the top.

2. Next, take the top level element of your HTML page, in our case the **<html>** element, call it the current element and add it as a child of the document.



3. For each element **nested** in the current element, add that element as a child of the current element in the DOM.

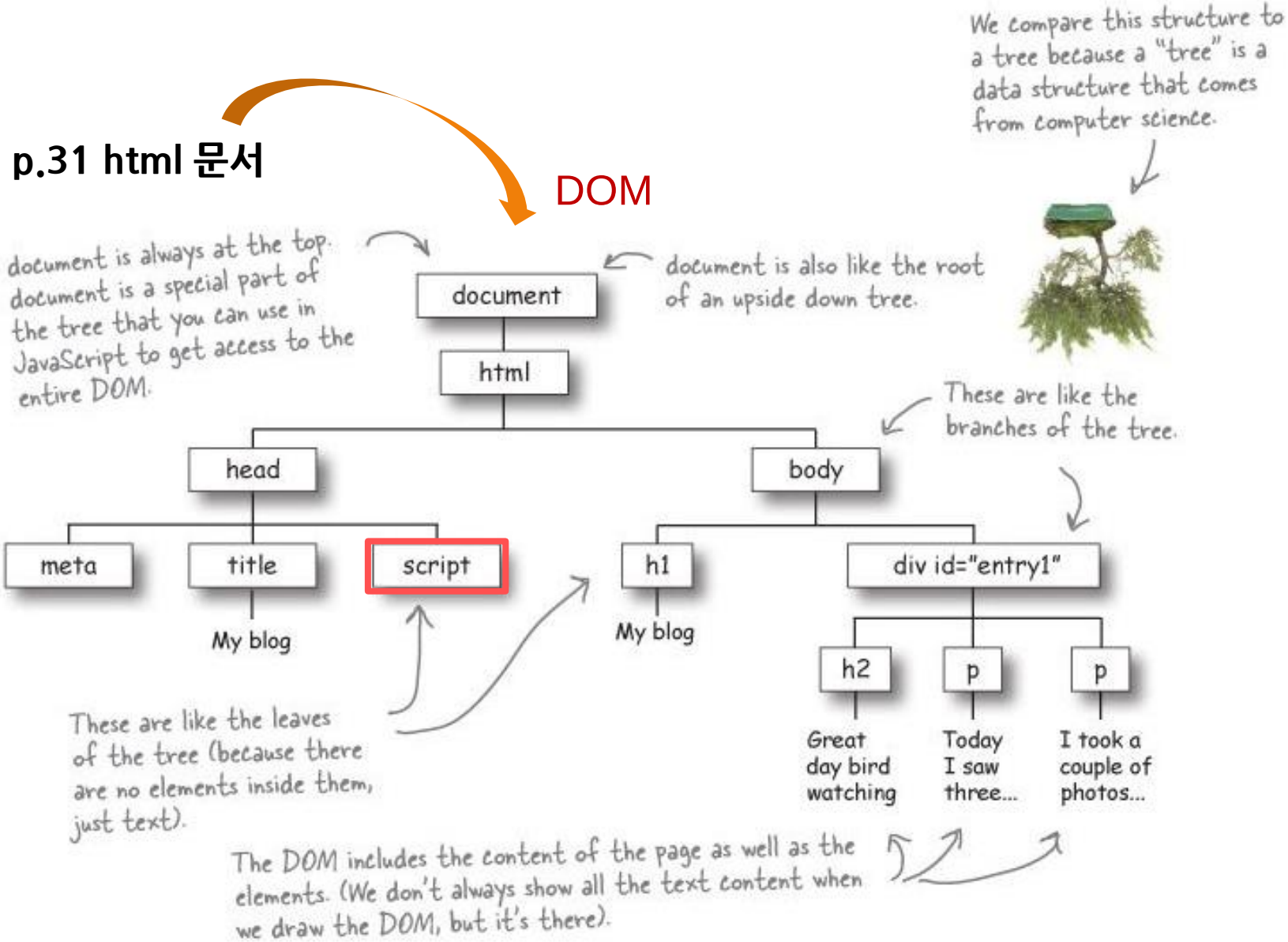


4. Return to (3) for each element you just added, and repeat until you are out of elements.

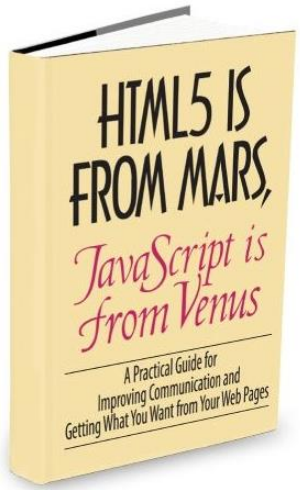

We've already fully
baked this DOM for you.
Turn the page to see the
finished DOM.

DOM의 매력은 코드에서 HTML의 구조와 내용에 접근하는 방식을 일관되게 제공한다는 것이다.

p.31 html 문서




Now that we have
a DOM we can examine
or alter it in any way we
want.



화성에서 온 HTML5

금성에서 온 JavaScript!!

How two totally different technologies hooked up.

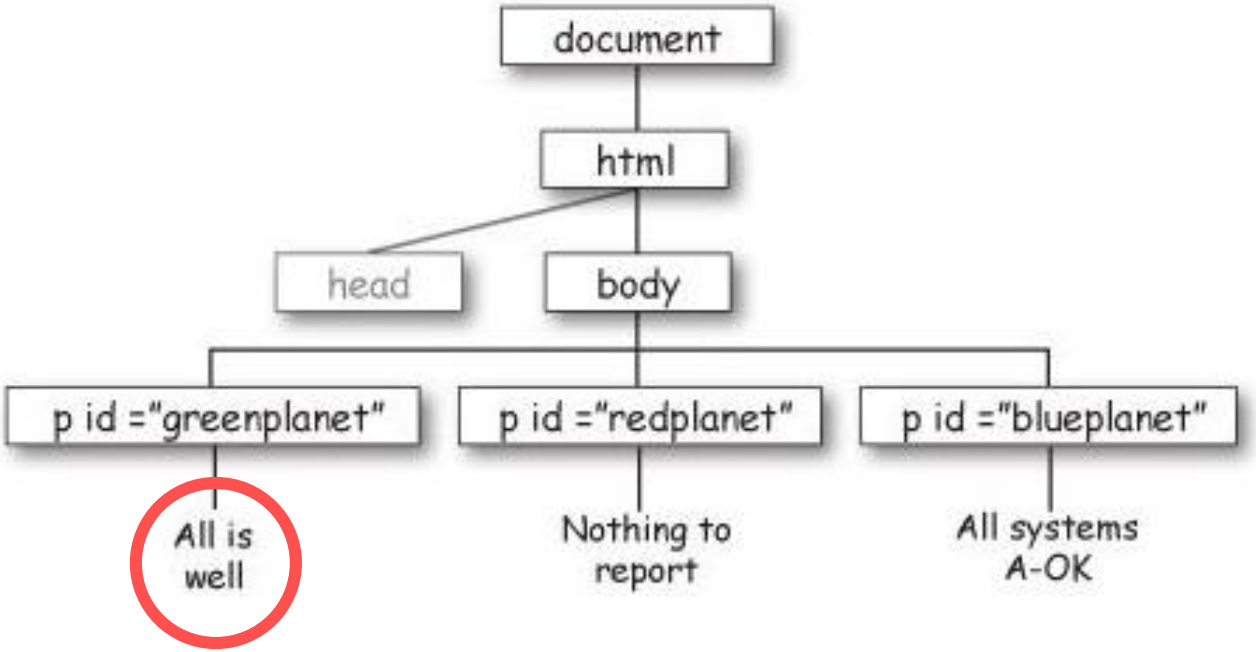
- HTML's DNA is made of declarative markup
- JavaScript is made of pure algorithmic genetic material

Are they so far apart they can't even communicate?

- Have something in common: the **DOM**
- **Through the DOM**, JavaScript can communicate with your page, and vice versa
- Is a little wormhole of sorts that allows JavaScript to get access to any element, and it's called getElementById

Let's start with a DOM

Here's a **simple DOM**; it's got a few HTML paragraphs, each with an id identifying it as the green, red or blue planet.



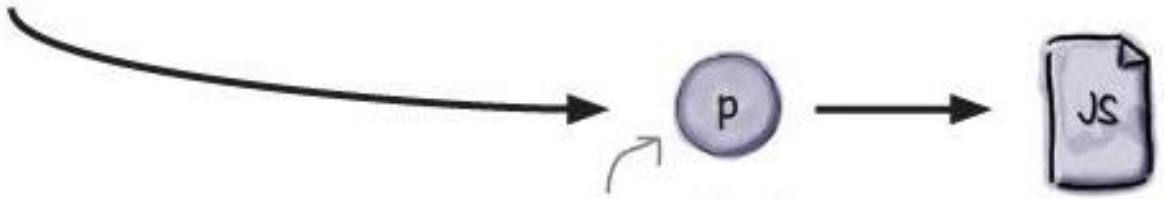
Now let's use JavaScript to make things more interesting

Let's say we want to **change** the greenplanet's text
from "**All is well**" to "**Red Alert: hit by phaser fire!**'
To do that we need the element with an id of greenplanet.

Remember the document represents the entire page in your browser and contains the complete DOM, so we can ask it to do things like find an element with a specific id.

Here we're asking the document to get us an element by finding the element that matches the given id.

```
document.getElementById("greenplanet");
```



getElementById("greenplanet") returns the paragraph element corresponding to "greenplanet"...

...and then the JavaScript code can do all sorts of interesting things with it..

Change its text to "Red Alert: hit by phaser fire!"

Once getElementById gives you an element, you're ready do something with it.

We're assigning the element to a variable named planet.

Here's our call to getElementById, which seeks out the "greenplanet" element and returns it.

```
var planet = document.getElementById("greenplanet");
```

And in our code we can now just use the variable planet to refer to our element.

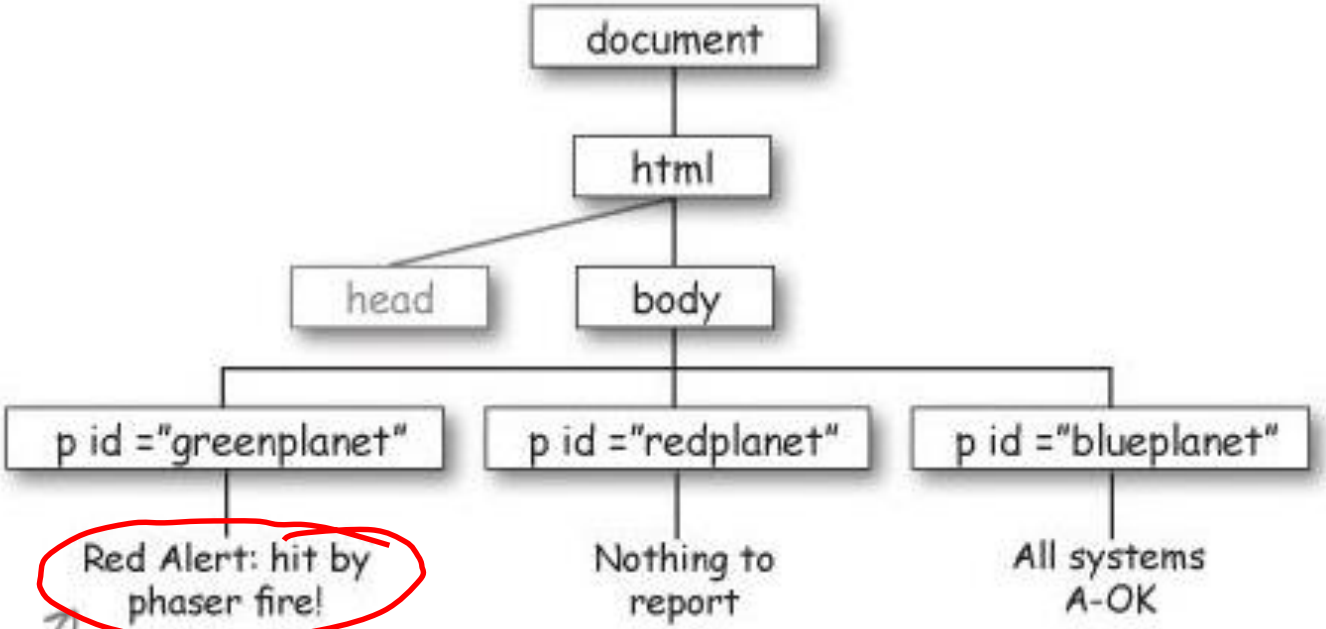
```
planet.innerHTML = "Red Alert: hit by phaser fire!";
```

We can use the innerHTML property of our planet element to change the content of the element.

We'll talk more about properties of elements shortly...

We change the content of the greenplanet element to our new text... which results in the DOM (and your page) being updated with the new text.

A little wormhole: getElementById



Any changes to the DOM are reflected in the browser's rendering of the page, so you'll see the paragraph change to contain the new content!

Test drive the planets: Uh Oh!

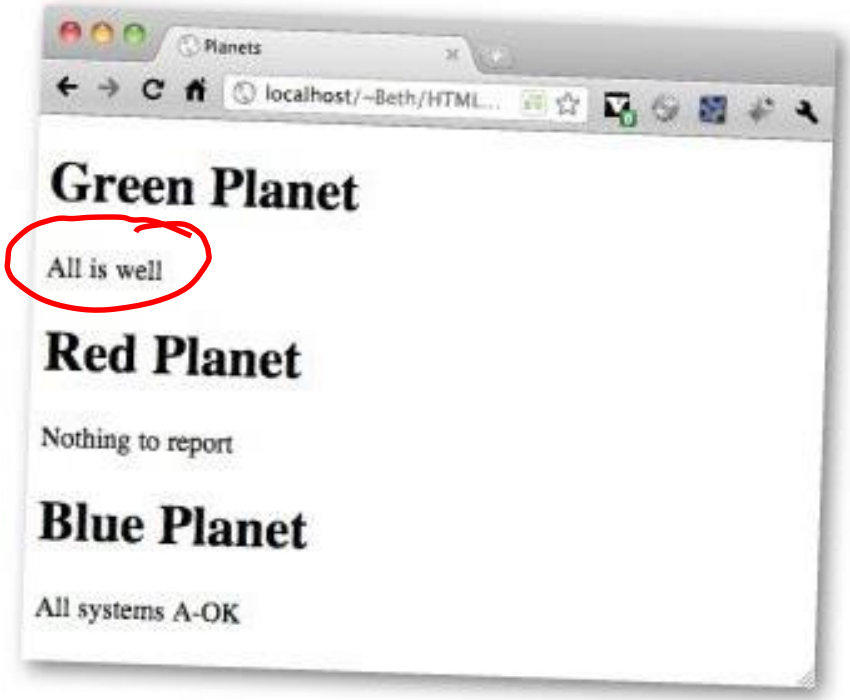
```
<!doctype html>
<html lang="en">
<head>
  <title>Planets</title>
  <meta charset="utf-8">
  <script>
    var planet = document.getElementById("greenplanet");
    planet.innerHTML = "Red Alert: hit by phaser fire!";
  </script>
</head>
<body>
  <h1>Green Planet</h1>
  <p id="greenplanet">All is well</p>
  <h1>Red Planet</h1>
  <p id="redplanet">Nothing to report</p>
  <h1>Blue Planet</h1>
  <p id="blueplanet">All systems A-OK</p>
</body>
</html>
```

We've added the JavaScript to the head of the page.

Just like you saw before, we're getting the <p> element with the id "greenplanet" and changing its content.

Here's the <p> element you're going to change with JavaScript.

실습과제 2-6


Planets
localhost/~Beth/HTML...
Green Planet
All is well
Red Planet
Nothing to report
Blue Planet
All systems A-OK

Oh yeah, we forgot to mention one thing

페이지가 완전히 로드된 후에 JavaScript 코드를 실행시키는 것이 의미가 있다. 이유?

- 브라우저가 먼저 페이지의 head 부분을 로드할 때 JavaScript 코드가 실행될 것이다. 즉, 페이지의 나머지 부분이 모두 로드되기 전에 코드가 실행된다.
  그러나, 아직 DOM은 완전하게 생성되지 않았을 것이다.

- DOM이 생성되지 않았다면 <p id="greenplanet"> 엘리먼트 역시 아직 존재하지 않는다.

What we need is a way to tell the browser

"Run my code after you've fully loaded in the page and created the DOM."

How do you tell the browser to execute your code only after it's loaded?

```
<script>
function init() {
    var planet = document.getElementById("greenplanet");
    planet.innerHTML = "Red Alert: hit by phaser fire!";
}

window.onload = init;
</script>
```

First, create a function named init and put your existing code in the function.

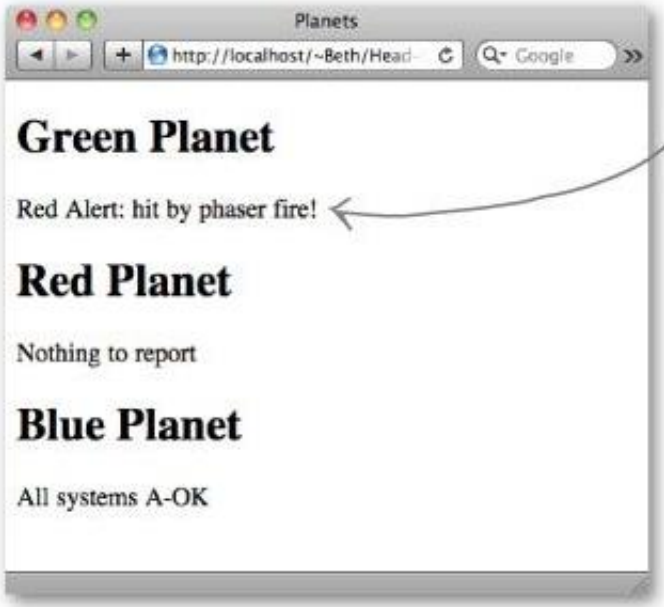
Notice that your code goes between an opening { and a closing }.

Here, we're setting the value of the window.onload property to the function name.

This says when the page is fully loaded, execute the code that is in init.

실습과제 2-7

Reload the page



Yes! Now we see the new content in the green planet <p> element. Isn't it great?

Well, what IS great is that now you know how to tell the browser to wait until the DOM has completely loaded before running code that accesses elements.

실습과제 2-8 Sharpen Your Pencil

百聞不如一打

Here's the HTML for the page.

```
<!doctype html>
<html lang="en">
<head>
  <title>My Playlist</title>
  <meta charset="utf-8">
  <script>
  ____________ addSongs() {
    var song1 = document._____________("________");
    var _____ = ______________________("________");
    var _____ = ________.getElementById("________");

    ________.innerHTML = "Blue Suede Strings, by Elvis Pagely";
    _________________ = "Great Objects on Fire, by Jerry JSON Lewis";
    song3.____________ = "I Code the Line, by Johnny JavaScript";
  }
  window.___________ = ____________;
  </script>
</head>
<body>
  <h1>My awesome playlist</h1>
  <ul id="playlist">
    <li id="song1"></li>
    <li id="song2"></li>
    <li id="song3"></li>
  </ul>
</body>
</html>
```

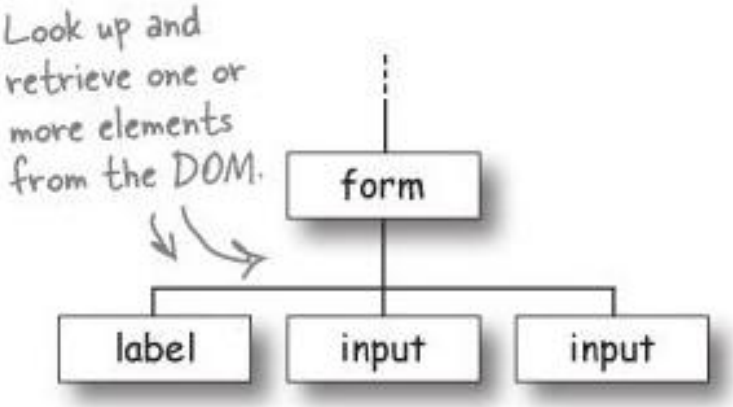
Here's our script. This code should fill in the list of songs below, in the <ul>.

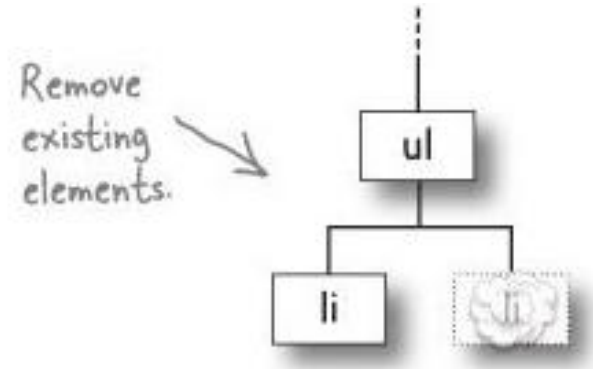
Fill in the blanks with the missing code to get the playlist filled out.

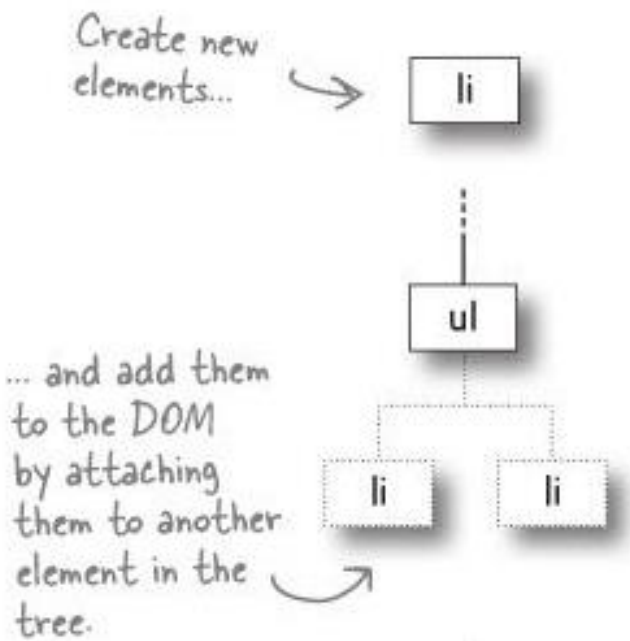
Here's the empty list of songs. The code above should add content to each <li> in the playlist.

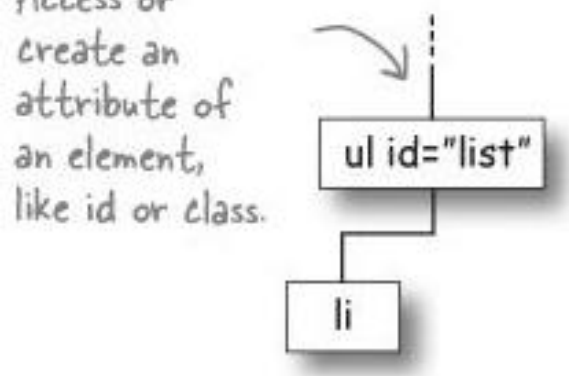
When you get the JavaScript working, this is what the web page will look like after you load the page.

So, what else is a DOM good for anyway





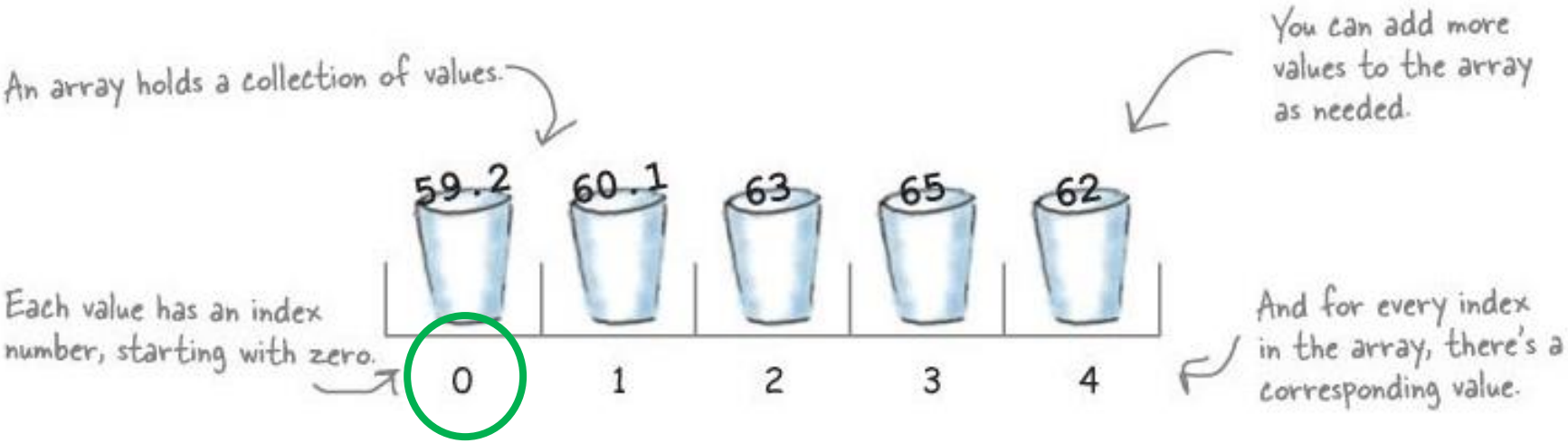




How to store multiple values in JavaScript

Array - JavaScript type that you'll use all the time

Let's say you wanted to store the names of thirty-two ice cream flavors or the item numbers of all the items in your user's shopping cart or maybe the outside temperature by the hour.



How to create an array

- 배열을 사용하기 전에 생성해야 한다
- 배열 자체를 변수에 할당해야 한다

Let's create the array with hourly temperatures:

Here's our variable for the array...

...and here's how we actually create a new empty array.

```
var tempByHour = new Array();
tempByHour[0] = 59.2;
tempByHour[1] = 60.1;
tempByHour[2] = 63;
tempByHour[3] = 65;
tempByHour[4] = 62;
```

We'll come back to this syntax in Chapter 4, but for now, just know that it creates a new array.

To add new values to the array, we just reference the index number of the array item, and give it a value.

Just like a variable in JavaScript you can assign any value (or type of value) to an array index.

The index.

=

```
var tempByHour = [59.2, 60.1, 63, 65, 62];
```

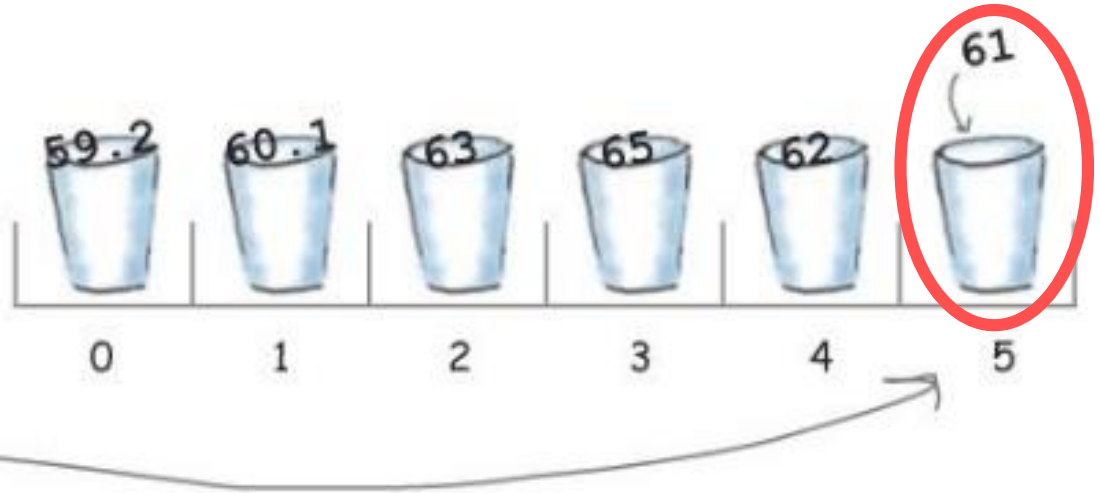
This creates the same array as above, just with a lot less code.

Adding another item to the array

언제든지 사용하지 않은 인덱스를 이용하여 새로운 아이템을 배열에 추가할 수 있다:

```
tempByHour[5] = 61;
```

By using a new index, we get a new item in the array.



Using your array items

배열변수의 인덱스 값을 참조함으로써 배열 아이템의 값을 얻어 올 수 있다:


```
var message = "The temperature at 5 was " + tempByHour[5];
alert(message);
```

To access the value of the temperature at index 5, we just reference the array at index 5.

Know the size of your array, or else

length라는 배열의 프로퍼티를 참조하여 **배열의 크기**를 얻어 올 수 있다:

```
var numItems = tempByHour.length;
```

We'll talk more about properties in the next chapter; for now, just know that every array has this length property that tells you the number of items in the array.

실습과제 2-9 Sharpen Your Pencil

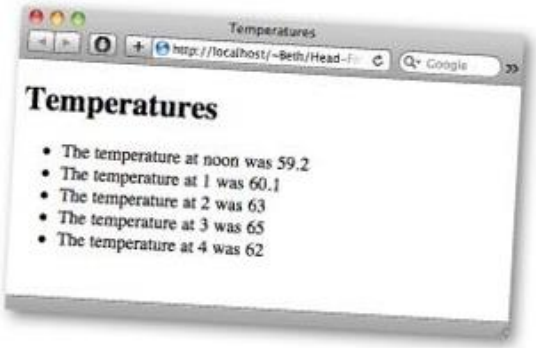
百聞不如一打

```
<!doctype html>
<html lang="en">
<head>
<title>Temperatures</title>
<meta charset="utf-8">
<script>
function showTemps() {
    var tempByHour = new ____________;
    tempByHour[0] = 59.2;
    tempByHour[1] = 60.1;
    tempByHour[2] = 63;
    tempByHour[3] = 65;
    tempByHour[4] = 62;
    for (var i = 0; i < ___________________; _____) {
        var theTemp = _____________[i];
        var id = "__________" + i;
        var li = document.__________________(id);
        if (i == ____) {
            li._______________ = "The temperature at noon was " + theTemp;
        } else {
            li.innerHTML = "The temperature at " + _______ + " was " + _________;
        }
    }
}
window.onload = showTemps;
</script>
</head>
<body>
<h1>Temperatures</h1>
<ul>
    <li id="temp0"></li>
    <li id="temp1"></li>
    <li id="temp2"></li>
    <li id="temp3"></li>
    <li id="temp4"></li>
</ul>
</body>
</html>
```

← Here's the HTML

Here's where we're combining loops and arrays. Can you see how we're accessing each item in the array using a variable index?

The code above will fill in each list item with a phrase with the temperature.



실습과제 2-10 Run the code below

百聞不如一打

```
<!doctype html>
<html lang="en">
<head>
    <title>Phrase-o-matic</title>
<meta charset="utf-8">
<style>
body {
    font-family: Verdana, Helvetica, sans-serif;
}
</style>
<script>
function makePhrases() {
    var words1 = ["24/7", "multi-Tier", "30,000 foot", "B-to-B", "win-win"];
    var words2 = ["empowered", "value-added", "oriented", "focused", "aligned"];
    var words3 = ["process", "solution", "tipping-point", "strategy", "vision"];

    var rand1 = Math.floor(Math.random() * words1.length);
    var rand2 = Math.floor(Math.random() * words2.length);
    var rand3 = Math.floor(Math.random() * words3.length);

    var phrase = words1[rand1] + " " + words2[rand2] + " " + words3[rand3];
    var phraseElement = document.getElementById("phrase");
    phraseElement.innerHTML = phrase;
}
window.onload = makePhrases;
</script>
</head>
<body>
    <h1>Phrase-o-Matic says:</h1>
    <p id="phrase"></p>
</body>
</html>
```

Q & A