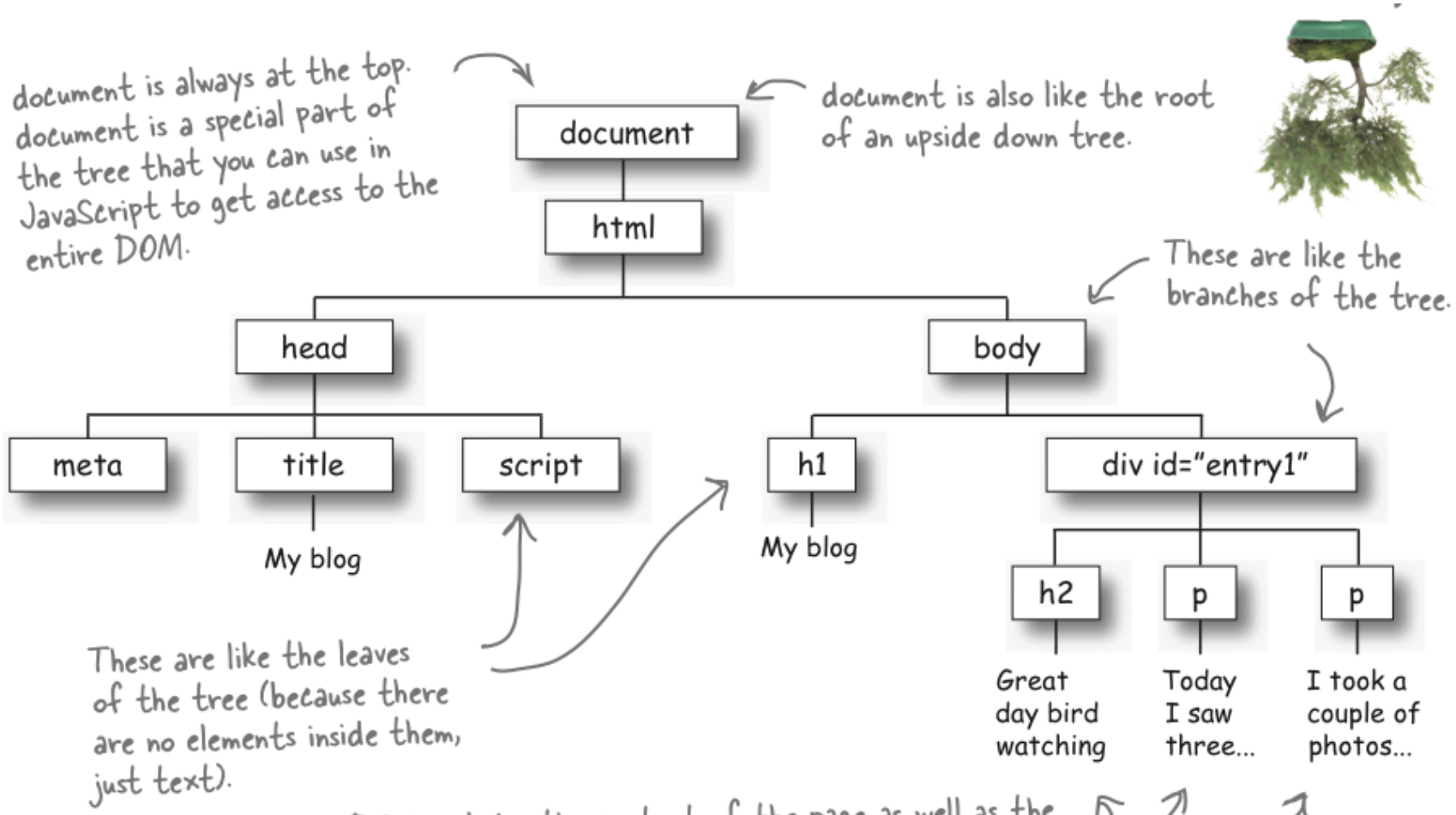
JavaScript

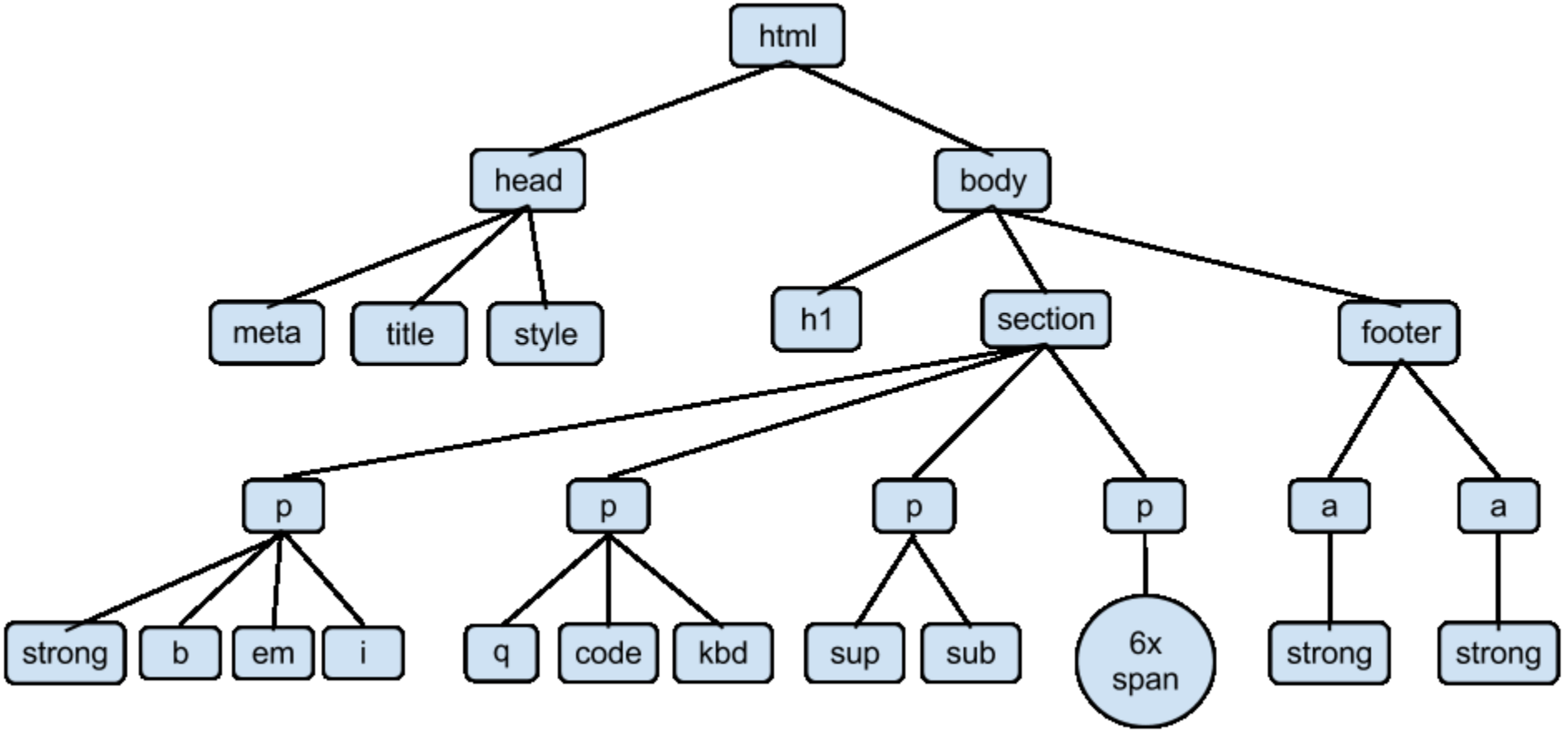
CHAPTER III. HTML DOM

Ths. Hồ Đức Lĩnh
Mail: duclinh47th@gmail.com
Phone: 0905 28 68 87

HTML DOM


document is always at the top. document is a special part of the tree that you can use in JavaScript to get access to the entire DOM.
document
document is also like the root of an upside down tree.
html
These are like the branches of the tree.
head
body
meta
title
script
h1
div id="entry1"
My blog
My blog
h2
p
p
Great day bird watching
Today I saw three...
I took a couple of photos...
These are like the leaves of the tree (because there are no elements inside them, just text).
The DOM includes the content of the page as well as the elements. (We don't always show all the text content when we draw the DOM, but it's there).

HTML DOM


html
head
body
meta
title
style
h1
section
footer
p
p
p
p
a
a
strong
b
em
i
q
code
kbd
sup
sub
6x
span
strong
strong

Finding HTML Elements

- Often, with JavaScript, you want to manipulate HTML elements.
- To do so, you have to find the elements first. There are several ways to do this:
  - Finding HTML elements by **id**
  - Finding HTML elements by **tag name**
  - Finding HTML elements by **class name**
  - Finding HTML elements by **CSS selectors**
  - Finding HTML elements by **HTML object collections**

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function() {*code*}	Adding event handler code to an onclick event

Finding HTML Element by **Id**

- The easiest way to find an HTML element in the DOM, is by using the element id.

```
<h2>Finding HTML Elements by Id</h2>

<p id="intro">Hello World!</p>
<p id="demo"></p>

<script>
    var myElement = document.getElementById("intro");
    document.getElementById("demo").innerHTML =
        "The text from the intro paragraph is " + myElement.innerHTML;
</script>
```

Here's the HTML.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Dr. Evel's Secret Code Page</title>
  </head>
  <body>
    <p id="code1">The eagle is in the</p>
    <p id="code2">The fox is in the</p>
    <p id="code3">snuck into the garden last night.</p>
    <p id="code4">They said it would rain</p>
    <p id="code5">Does the red robin crow at</p>
    <p id="code6">Where can I find Mr.</p>
    <p id="code7">I told the boys to bring tea and</p>
    <p id="code8">Where's my dough? The cake won't</p>
    <p id="code9">My watch stopped at</p>
    <p id="code10">barking, can't fly without umbrella.</p>
    <p id="code11">The green canary flies at</p>
    <p id="code12">The oyster owns a fine</p>
    <script src="code.js"></script>
  </body>
</html>
```

Notice that each paragraph is identified by an id.

Here's the JavaScript...

Let's start with a DOM

document is a global object.

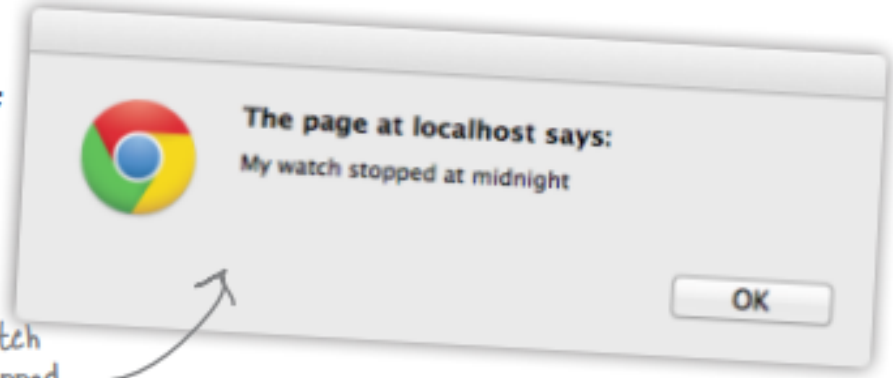
And getElementById is a method.

Make sure you get the case right on the letters in the method name getElementById, otherwise it won't work!

```
var access =
  document.getElementById("code9");
var code = access.innerHTML;
code = code + " midnight";
alert(code);
```

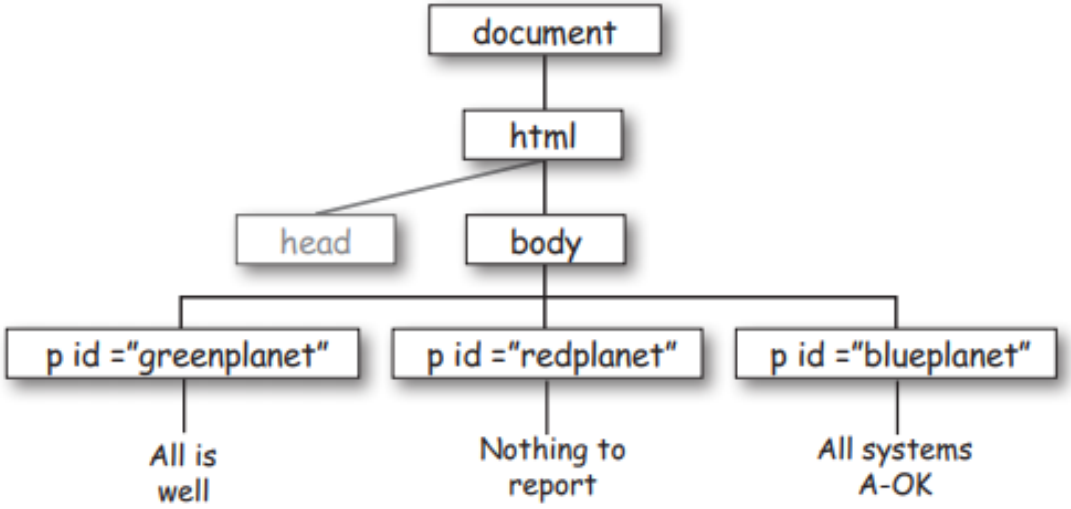
And look, we have dot notation, this looks like an object with an innerHTML property.

```
var access =
  document.getElementById("code9");
var code = access.innerHTML;
code = code + " midnight";
alert(code);
```



So we add " midnight" to "My watch stopped at" to get "My watch stopped at midnight" and then put up an alert to display this code.

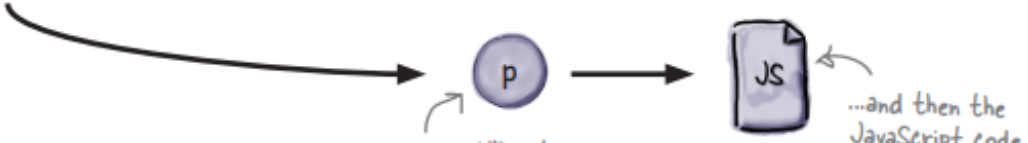
Let's start with a DOM



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.

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");
```

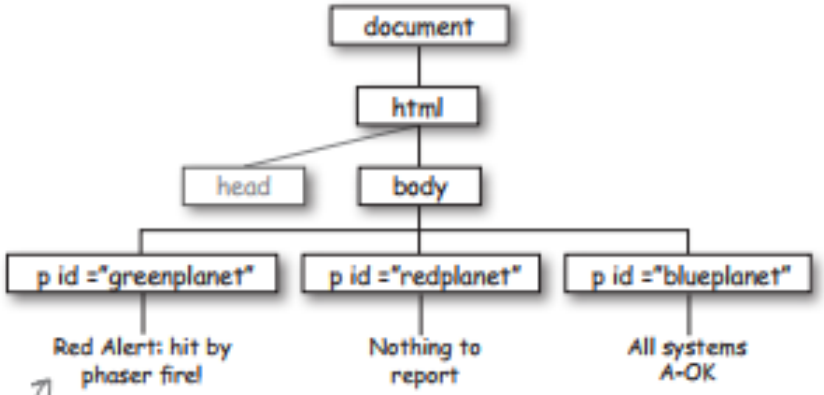
And in our code we can now just use the variable planet to refer to our element.

```
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.

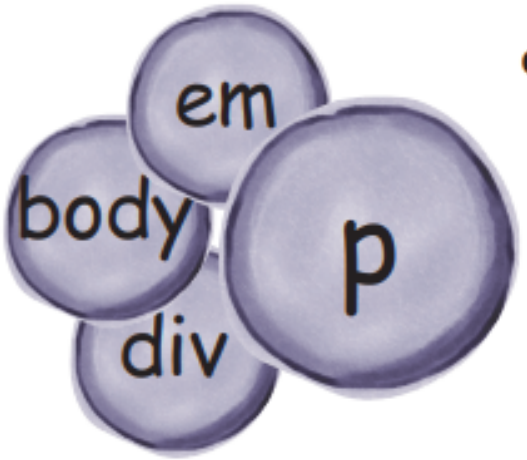
We'll talk more about properties of elements shortly...

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.



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!

What, exactly, am I getting from the DOM?



Things you can do with an element object.

Get the content (text or HTML).

Change the content.

Read an attribute.

Add an attribute.

Change an attribute.

Remove an attribute.

The planet variable contains an element object—the element object that is the "greenplanet" <p> element.

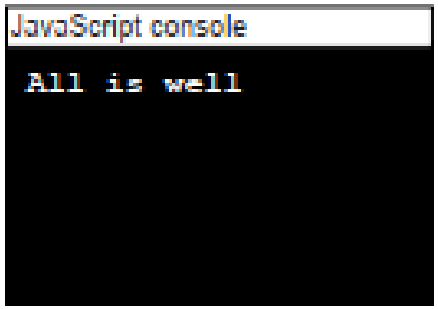
```
var planet = document.getElementById("greenplanet");

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

We can use the innerHTML property of the element object to change the content of the element!

Finding your inner HTML

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

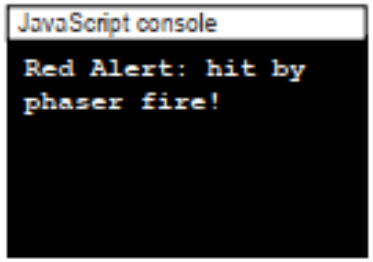
We're just passing the planet.innerHTML property to console.log to log to the console.



The content of the innerHTML property is just a string, so it displays just like any other string in the console.

```
var planet = document.getElementById("greenplanet");
planet.innerHTML = "Red Alert: hit by phaser fire!";
console.log(planet.innerHTML);
```

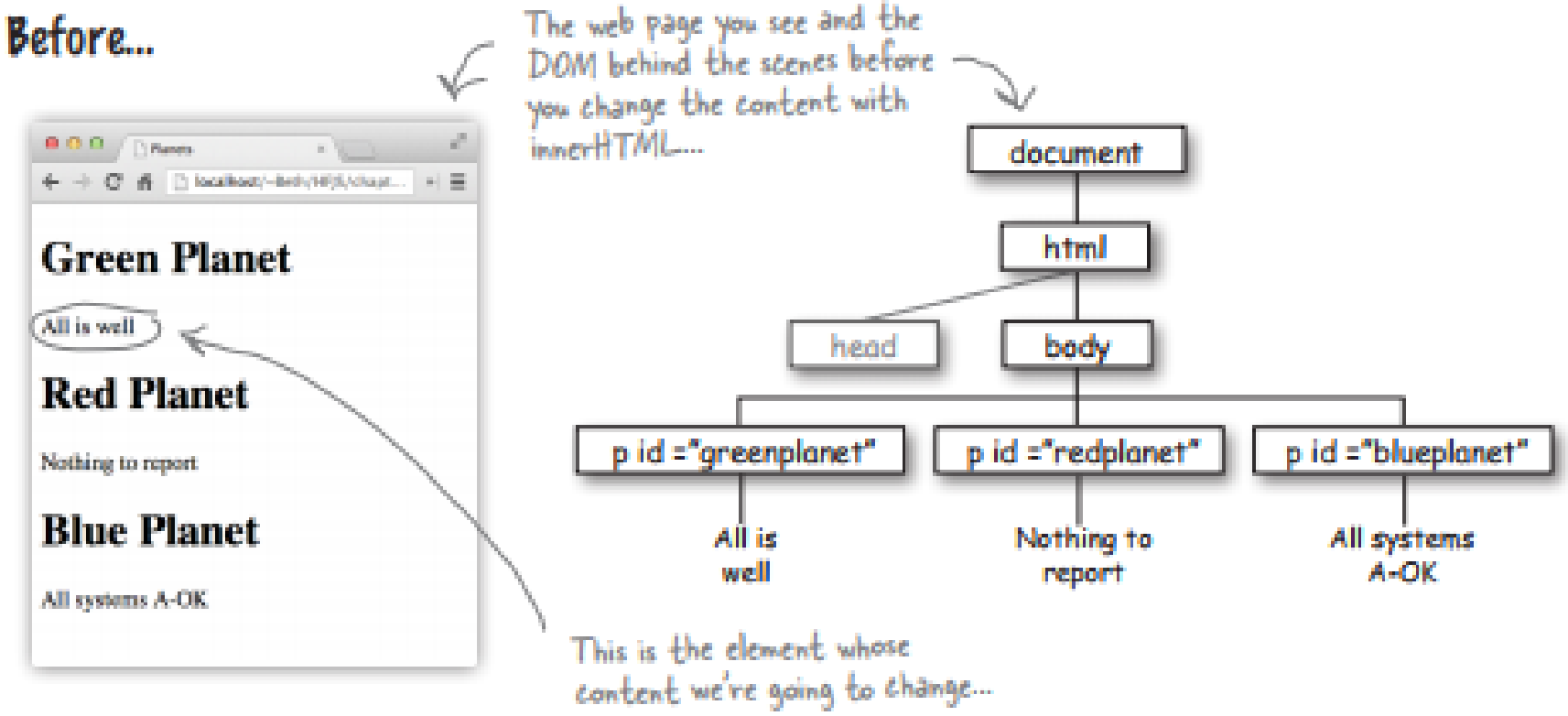
Now we're changing the content of the element by setting its innerHTML property to the string "Red Alert: hit by phaser fire!"



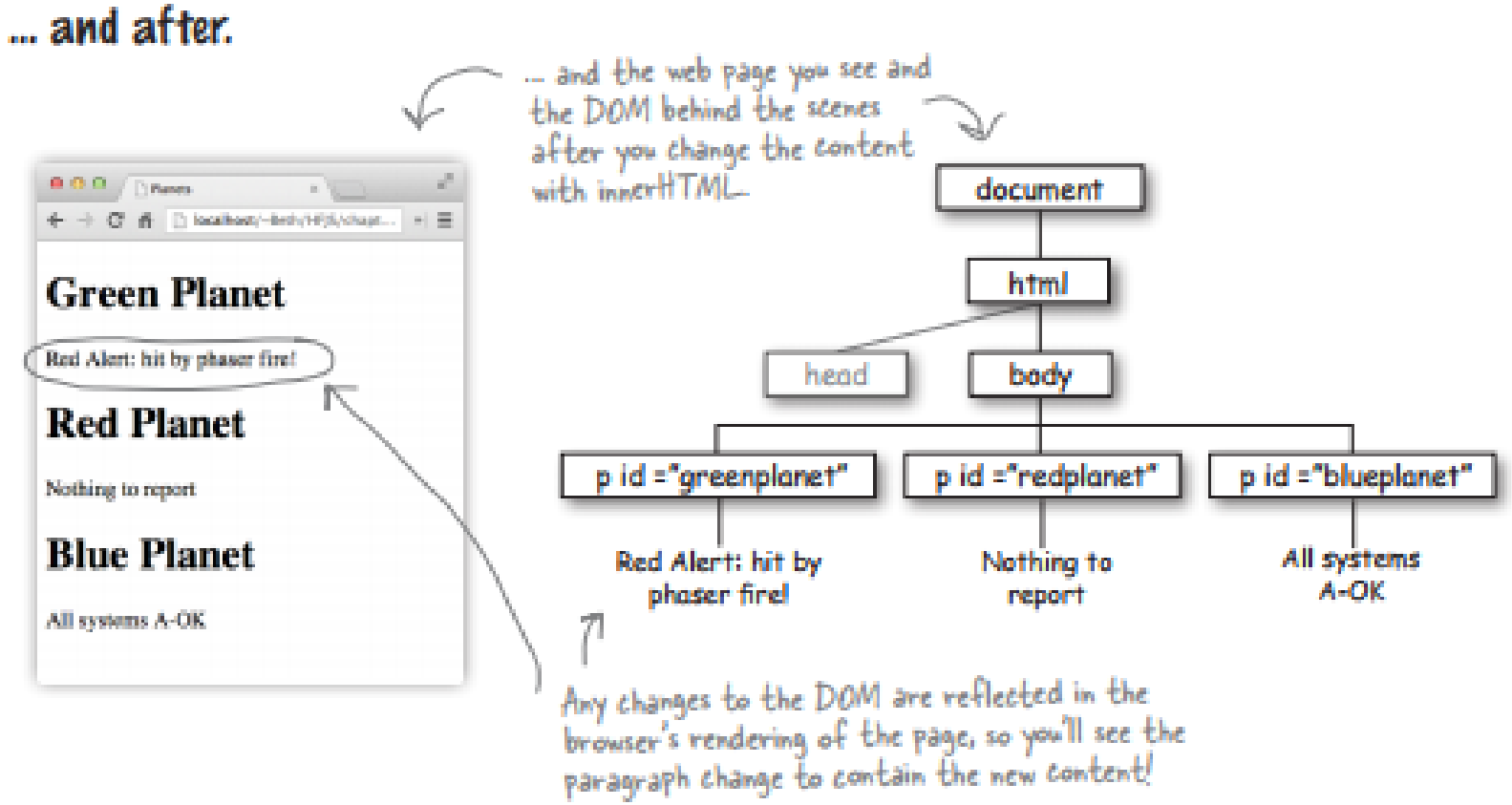
So when we log the value of the innerHTML property to the console we see the new value.

And the web page changes too!

What happens when you change the DOM?



What happens when you change the DOM?



Don’t even think about running my code until the page is fully loaded!

You can call this function anything you want, but it's often called init by convention.

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

Here's the code we had before, only now it's in the body of the init function.

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

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

Here, we're assigning the function init to the window.onload property. Make sure you don't use parentheses after the function name! We're not calling the function; we're just assigning the function value to the window.onload property.

Sharpen your pencil

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.

```
<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>
```

```
<body>
  <h1>My awesome playlist</h1>
  <ul id="playlist">
    <li id="song1"></li>
    <li id="song2"></li>
    <li id="song3"></li>
  </ul>
</body>
```

How to set an attribute with setAttribute?

We take our element object.

```
planet.setAttribute("class", "redtext");
```

Note if the attribute doesn't exist a new one will be created in the element.

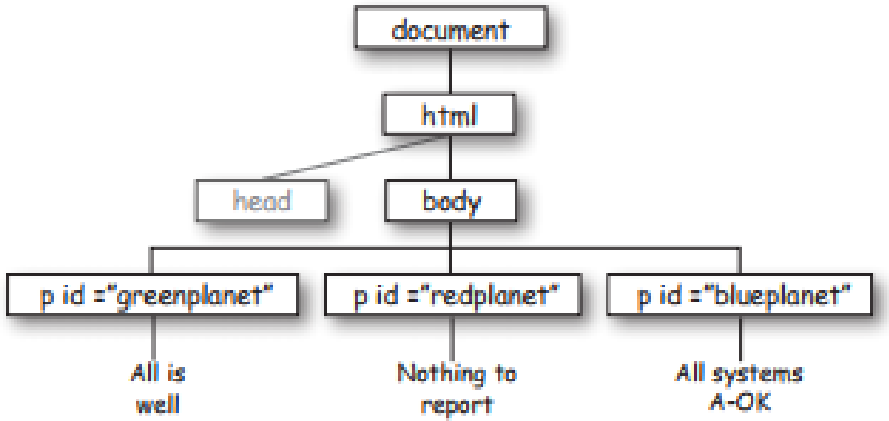
And we use its setAttribute method to either add a new attribute or change an existing attribute.

The method takes two arguments, the name of the attribute you want to set or change...

... and the value you'd like to set that attribute to.

Before...

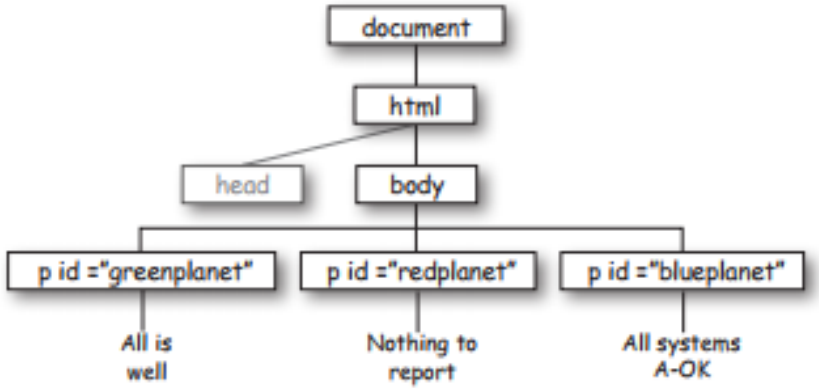
Here's the element before we call the setAttribute method on it. Notice this element already has one attribute, id.



How to set an attribute with setAttribute?

Before...

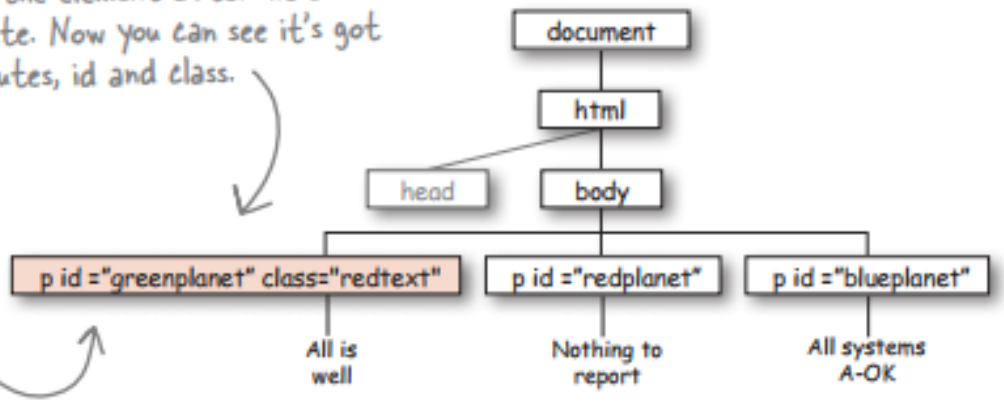
Here's the element before we call the setAttribute method on it. Notice this element already has one attribute, id.



And After

And here's the element after we call setAttribute. Now you can see it's got two attributes, id and class.

Remember, when we call the setAttribute method, we're changing the element object in the DOM, which immediately changes what you see displayed in the browser.



More fun with attributes!

Get a reference to the element with getElementById, then use the element's getAttribute method to get the attribute.

Pass in the name of the attribute you want the value of.

```
var scoop = document.getElementById("raspberry");
var altText = scoop.getAttribute("alt");
console.log("I can't see the image in the console,");
console.log(" but I'm told it looks like: " + altText);
```

Modifying Attributes

Method	Description	Example
`hasAttribute()`	Returns a `true` or `false` boolean	`element.hasAttribute('href');`
`getAttribute()`	Returns the value of a specified attribute or `null`	`element.getAttribute('href');`
`setAttribute()`	Adds or updates value of a specified attribute	`element.setAttribute('href', 'index.html');`
`removeAttribute()`	Removes an attribute from an element	`element.removeAttribute('href');`

Modifying Classes

Method/Property	Description	Example
`className`	Gets or sets class value	`element.className;`
`classList.add()`	Adds one or more class values	`element.classList.add('active');`
`classList.toggle()`	Toggles a class on or off	`element.classList.toggle('active');`
`classList.contains()`	Checks if class value exists	`element.classList.contains('active');`
`classList.replace()`	Replace an existing class value with a new class value	`element.classList.replace('old', 'new');`
`classList.remove()`	Remove a class value	`element.classList.remove('active');`

More fun with attributes! (Cont ..)

```
<script>
    function init(){
        var scoop = document.getElementById("myImg");
        var altText = scoop.getAttribute("alt");
        if (altText != null) {
            console.log("I'm told it looks like: "+altText);
        } else {
            var newAltText = scoop.setAttribute('alt',"This's new alt text for
            image");
            var newTitle = scoop.setAttribute("title","Here's new Title for
            Image");
            console.log(newAltText);
        }
    }
    window.onload = init;
</script>
```

```
<img id="myImg" src="images/raspberry.jpg">
```

More fun with attributes! (Cont ..)

```
<style>
    .bluetext{
        color: blue;
        font-size: 20px;
    }
</style>
<script type="text/javascript">
    function init(){
        var blue_color = document.getElementById("blue");
        blue_color.innerHTML = "Blue Color for this Tag";
        blue_color.setAttribute("class","bluetext");
    }
    window.onload = init;
</script>
```

```
<body>
    <p id="blue">Blue Text</p>
    <p id="red">Red Text</p>
    <p id="yellow">Yellow Text</p>
</body>
```

Blue Color for this Tag

Red Text

Yellow Text

CREATES THE HTML ELEMENT SPECIFIED BY *TAGNAME*

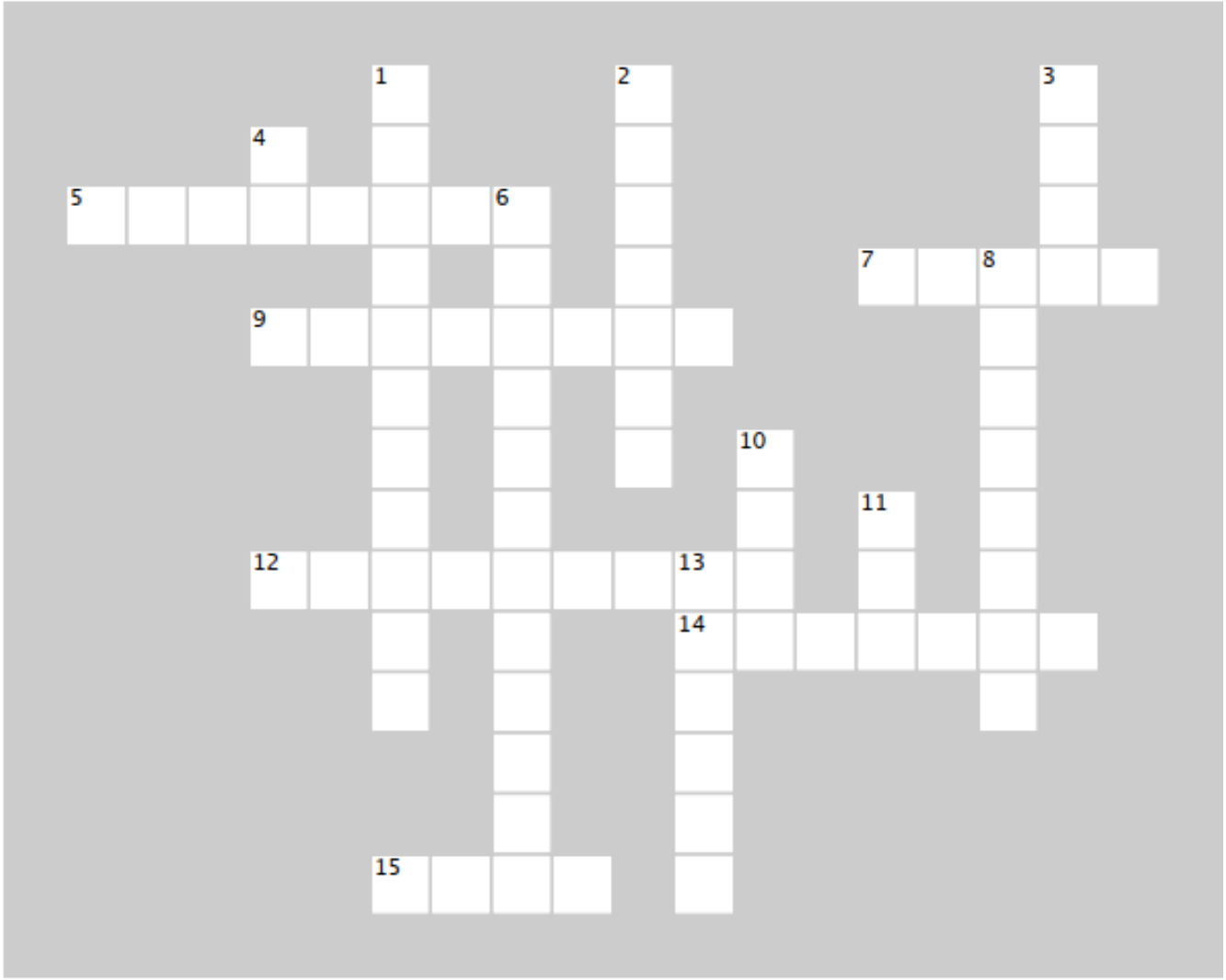
This creates a new `<div>` and inserts it before the element with the ID "`div1`".

```
<body>
    <div id="div1">The text above has been created dynamically.</div>
    <script>
            document.body.onload = addElement;
            function addElement(){
                //create a new Div element
                var newDiv = document.createElement("div");
                //and give it some content
                var newContent = document.createTextNode("Hi there and greetings !");
                //add the text node to the newly created div
                newDiv.appendChild(newContent);
                //add the newly created element and its content into the Dom
                var currentDiv = document.getElementById("div1");
                document.body.insertBefore(newDiv,currentDiv);
            }
    </script>
</body>
```

Hi there and greetings !
The text above has been created dynamically.

JavaScript cross

Load the DOM into your brain with this puzzle.



ACROSS

5. Functions that handle events are known as event _____.

7. Dr. Evel's passcode clue was in the element with the id _____.

9. Assign a _____ to the window.onload property to handle the load event.

12. Use the element object's property, _____, to change the HTML inside an element.

14. The setAttribute method is a method of an _____ object.

15. The DOM is shaped like a _____.

DOWN

1. Which planet gets hit by phaser fire?

2. Use the _____ to see if you have errors in your code.

3. It's important to make sure the ____ is completely loaded before using code to get or change elements in the page.

4. The getElementById method gets an element by its ___.

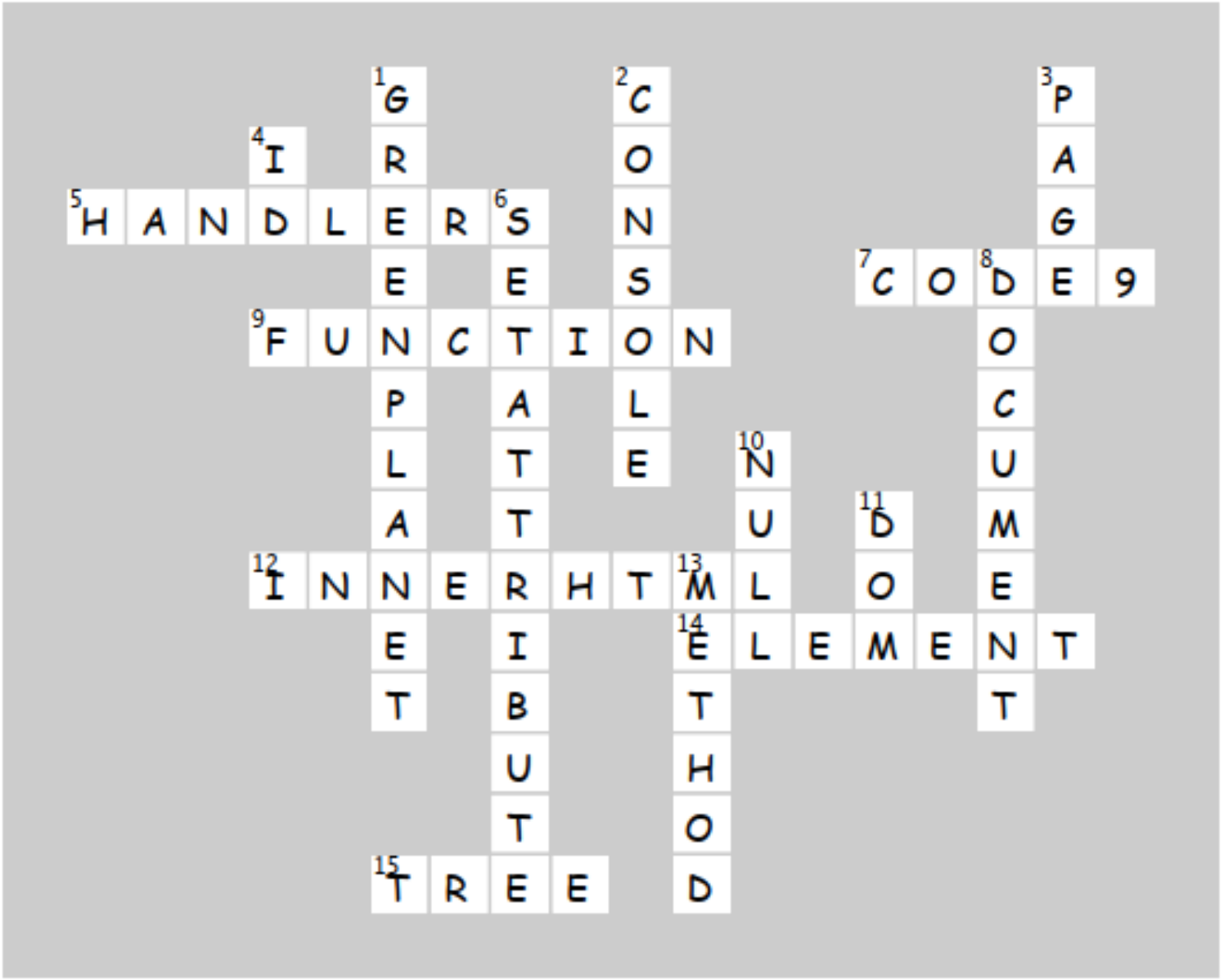
6. Change the class of an element using the _____ method.

8. The _____ object is always at the top of the DOM tree.

10. It's a good idea to check for ____ when using getElementById.

11. When you load a page into the browser, the browser creates a ____ representing all the elements and content in the page.

13. getElementById is a _____ of the document object.


1 G
2 C
3 P
4 I R O A
5 HANDLER 6 S N G
E E S 7 C O 8 D E 9
9 FUNCTION O
P A L C
L T E 10 N U
A T U 11 D M
12 INNERHT 13 M L O E
E I 14 ELEMENT
T B T T
U H
T O
15 TREE D