



JavaScript

More on using functions
and the DOM



Opening Exercise

1. Write a function ***setText*** that takes a CSS selector and a string as arguments, finds the element using the provided selector and changes its text content to the provided string.
2. Test your function on any web page using the console
3. Console usage hint:
 - When you hit ENTER, the console immediately evaluates what you have typed.
 - This may be inconvenient if you need to type more than one lines before any evaluation takes place - like when typing in a function definition!
 - Use SHIFT-ENTER (hold down the SHIFT key and press ENTER) to use multi-line mode: you will move to the next line without the previous one being evaluated. When you are done - hit ENTER (as usual).

My Solution

```
function setText(selector, text) {  
    //find the HTML element  
    var element = document.querySelector(selector);  
    //change the value of its innerText property  
    element.innerText = text;  
}  
  
//now test it!  
setText("p", "this is my new text!");
```

My Solution

```
function setText(selector, text) {  
    //find the HTML element  
    var element = document.querySelector(selector);  
    //change the value of its innerText property  
    element.innerText = text;  
}  
  
//now test it!  
setText("p", "this is my new text!");
```

My Solution

```
function setText(selector, text) {  
    //find the HTML element  
    var element = document.querySelector(selector);  
    //change the value of its innerText property  
    element.innerText = text;  
}  
  
//now test it!  
setText("p", "this is my new text!");
```

My Solution

```
function setText(selector, text) {  
    //find the HTML element  
    var element = document.querySelector(selector);  
    //change the value of its innerText property  
    element.innerText = text;  
}
```

```
//now test it!
```

```
setText("p", "this is my new text!");
```

...on a sidenote...

- The green text was comments
- Comments are ignored by JavaScript: their purpose is to make our code more understandable

- Comments can be:

```
//one-line comments
```

but also

```
/* they can
```

```
    span many
```

```
    lines */
```

- Use them sparingly: whenever you can use good variable names - then your code will be self-explanatory

Review: Functions

- What is a function?
 - is a named piece of code
 - that groups statements together
 - and makes code reusable
- How do we use a function? *(hint: 2 steps)*

1. Declare the function

```
function wakeMeUp() {  
    alert("Wake up, dude!");  
}
```

2. Call the function

```
wakeMeUp();
```


Applying functions: solution 0



```
//do the calculations
var wives = 7;
var sacks = 7 * wives;
var cats = 7 * sacks;
var kittens = 7 * cats;
var total = wives + sacks + cats + kittens;

//display the result as the text of a specific HTML element (with id="result")
var target = document.querySelector("#result");
target.innerText = total;
```

*As I was going to St. Ives, I met a man with seven wives.
Every wife had seven sacks, and every sack had seven cats, and every cat had seven kittens.
Kittens, cats, sacks, and wives, how many were going to St. Ives?*

Applying functions: solution 1



```
//place everything inside a function
function countItemsAndDisplayResult() {
    //do the calculations
    var wives = 7;
    var sacks = 7 * wives;
    var cats = 7 * sacks;
    var kittens = 7 * cats;
    var total = wives + sacks + cats + kittens;

    //display the result as the text of a specific HTML element (with id="result")
    var target = document.querySelector("#result");
    target.innerText = total;
}

//call the function
countItemsAndDisplayResult();
```

Applying functions: solution 2



//place calculations inside a function and return the result

```
function countItems() {  
    var wives = 7;  
    var sacks = 7 * wives;  
    var cats = 7 * sacks;  
    var kittens = 7 * cats;  
    return wives + sacks + cats + kittens;  
}
```

//call the function once and assign the result to a variable

```
var items = countItems();
```

//display the result anywhere you like, as many times as you like

```
document.querySelector("#result").innerText = items;
```

```
document.querySelector("h1").innerText = items;
```

```
alert("total item count = " + items);
```

Applying functions: solution 3



```
//add a parameter to the function definition
```

```
function countItems(times) {  
    var wives = times;  
    var sacks = times * wives;  
    var cats = times * sacks;  
    var kittens = times * cats;  
    return wives + sacks + cats + kittens;  
}
```

```
//call the function for as many quantities as you like
```

```
var itemsA = countItems(7);  
var itemsB = countItems(8);  
var itemsC = countItems(42);
```

```
//display the results anywhere you like, as many times as you like
```

```
document.querySelector("#result").innerText = itemsA;  
document.querySelector("h1").innerText = itemsB;  
alert("total items = " + itemsC);
```

Functions that take arguments: a closer look

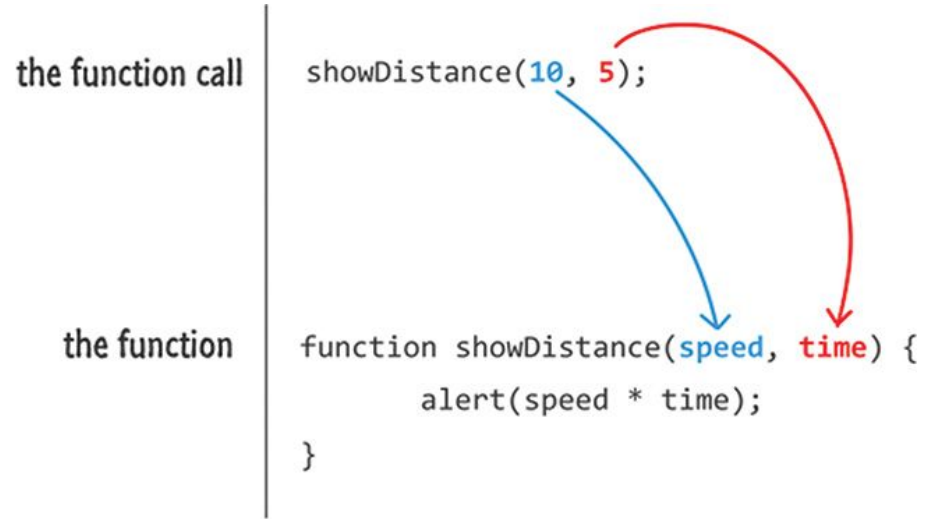
- How do we know if a function takes arguments?
- We look at the function definition to see if it has any **parameters**
- When we call the function, we pass these arguments to the function:
 - **10** and **5** are **arguments**
 - that we pass to the function
 - **speed** and **time** are **parameters**
 - that we list in our function definition

the function call

```
showDistance(10, 5);
```

the function

```
function showDistance(speed, time) {  
    alert(speed * time);  
}
```



The diagram illustrates the mapping between function arguments and parameters. A vertical line separates the function call from the function definition. A blue arrow points from the argument '10' in the function call to the parameter 'speed' in the function definition. A red arrow points from the argument '5' in the function call to the parameter 'time' in the function definition.

Functions that take arguments: a closer look

Our arguments can be represented by:

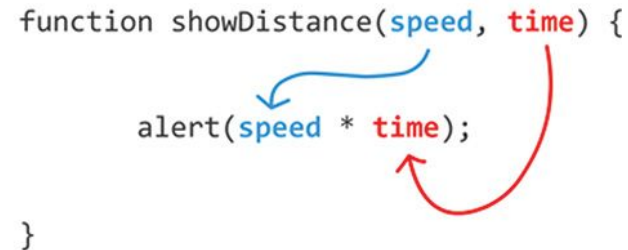
- values
 - `showDistance(10, 5)`
- variables holding values:
 - `var myDistance = 100;`
`var myTime = 5;`
`showDistance(myDistance, myTime);`
- expressions that are then evaluated and produce values
 - `showDistance(20 + 80, 10 / 2);`
- or any combination of the above:
 - `showDistance(myDistance + 80, 5);`

Functions that take arguments: a closer look

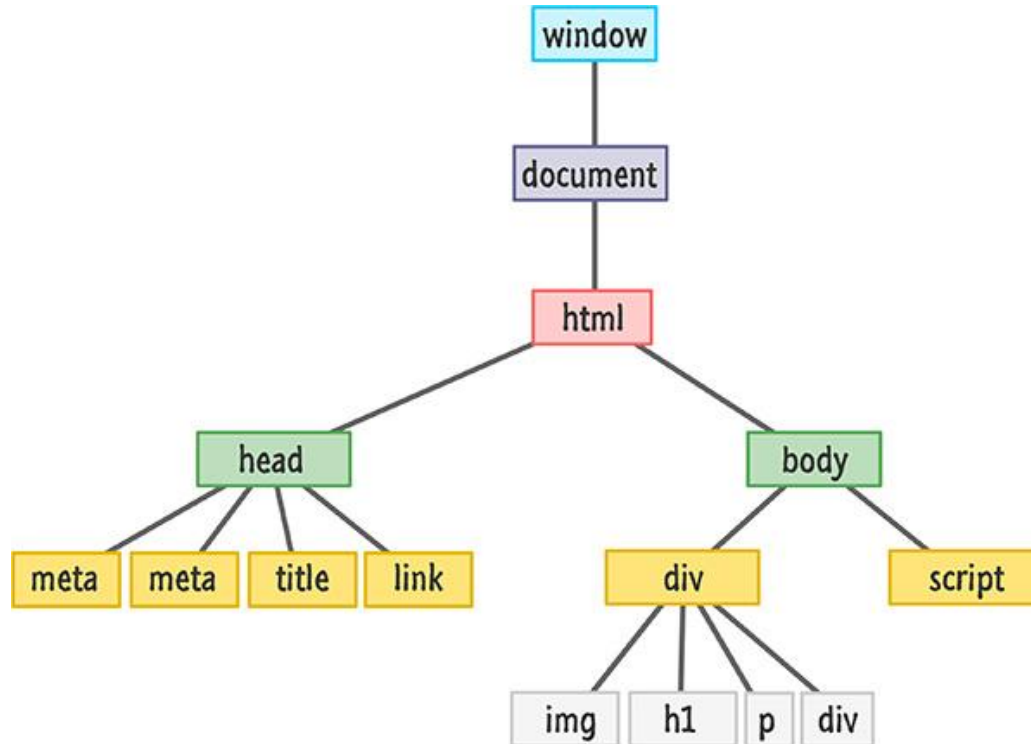
- How do we specify what a function does with our arguments?
 - We use the names of the parameters in the function definition as variable names -
 - - that exist ONLY inside the body of the function
 - JavaScript doesn't care what names we choose for our parameters -
 - - but choose good names so our code is readable and self-explanatory
- What happens to our arguments inside the function?
 - Our arguments are copied into variables -
 - - that exist only while the function executes

the function

```
function showDistance(speed, time) {  
    alert(speed * time);  
}
```

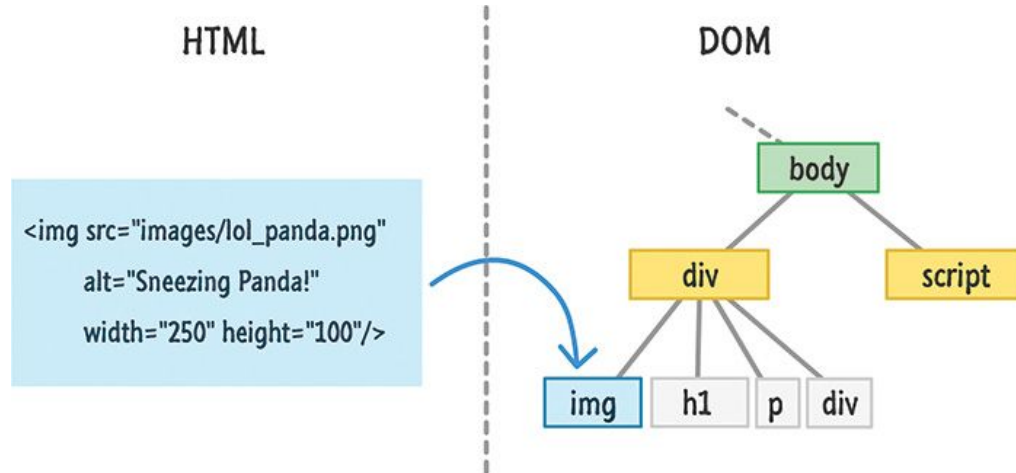


How does this apply to the DOM?



How does this apply to the DOM?

- The **D**ocument **O**bject **M**odel (DOM) tree is an object
 - that is a model of the page HTML where everything is an object

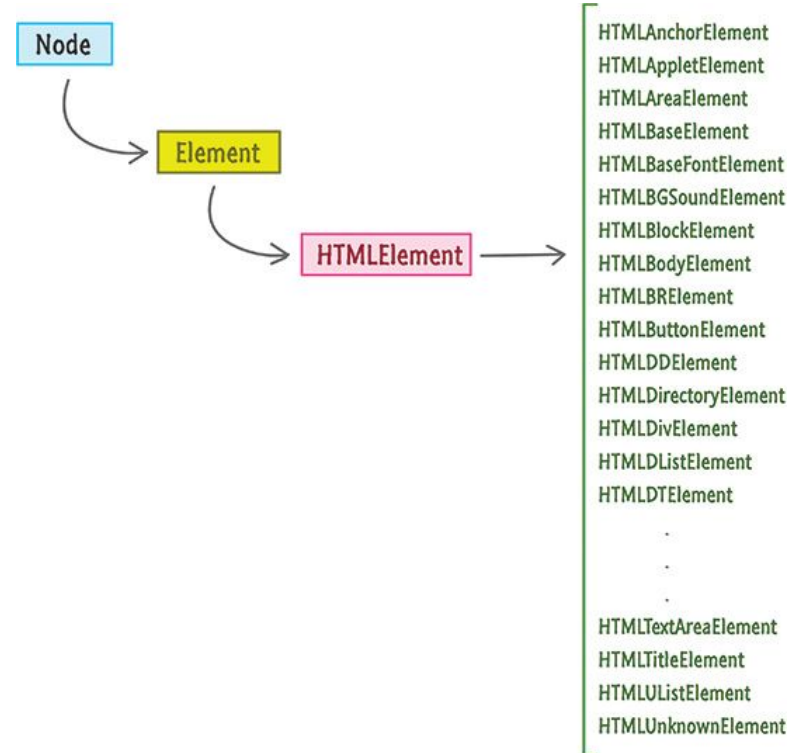


How does this apply to the DOM?

- An **object** is an abstraction *(of any entity we can describe)*
- An object provides access to methods and properties
 - we say that method X or property Y are part of the definition of object Z
- A **property** is a characteristic of an object that we may be able to access and modify *(or **get** and **set**)*
- A **method** is a property that we can call - i.e., it's a function that belongs to an object!
- Any function we create implicitly becomes part of the **window** object definition *(and so are the `alert()` and `prompt()` functions we've used)*

How does this apply to the DOM?

- Everything is an object
- Objects give us access to methods and properties
- We can use this to manipulate almost anything on our web page!



...so, what can we do with the DOM?

- Access any element

```
var element = document.querySelector("p")
```

- or group of elements (more on this next week)

```
var elementArray = document.querySelectorAll("p")
```

- access or modify the element's text content:

```
element.innerText = "new text";
```

- or its HTML content:

```
element.innerHTML = "new <b>text</b>";
```

...so, what can we do with the DOM?

- Access an element attribute

```
element.getAttribute("src");
```

- Modify an elements attribute

```
element.setAttribute("src", "newImage.jpg");
```

- Access an element's ID or class value

```
element.id;
```

```
element.className
```

...so, what can we do with the DOM?

- Set an element's style!

```
element.style.background = "red";
```

- or better: use the classList property to do the same

```
element.classList.add(someClass);
```

```
element.classList.remove(someClass);
```

```
element.classList.toggle(someClass);
```

```
element.classList.contains(someClass);
```

- ...and we'll practice this on our next lab!