sourcemind

JAVASCRIPT BASICS

FUNCTIONS
DOM API

Functions

A JavaScript function is a block of code designed to perform a particular task.

Functions are the main “building blocks” of the program. They allow the code to be called many times without repetition.

Function Declaration

```
function showMessage() {
  alert( 'Hello everyone!' );
}
```

```
function name(parameter1, parameter2, ... parameterN) {
  ...body...
}
```

Default values

```
function showMessage(from, text = "no text given") {
  alert( from + ": " + text );
}
```

```
function showMessage(text) {
  // ...

  if (text === undefined) { // if the pa
    text = 'empty message';
  }

  alert(text);
}

showMessage(); // empty message
```

Function Return

```
let x = myFunction(4, 3);   // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

A function with an empty `return` or without it returns `undefined`

Never add a newline between `return` and the value

For a long expression in `return`, it might be tempting to put it on a separate line, like this:

```
return
 (some + long + expression + or + whatever * f(a) + f(b))
```

That doesn't work, because JavaScript assumes a semicolon after `return`. That'll work the same as:

```
return;
 (some + long + expression + or + whatever * f(a) + f(b))
```

Function naming

- `"get…"` – return a value,
- `"calc…"` – calculate something,
- `"create…"` – create something,
- `"check…"` – check something and return a boolean, etc.

One function – one action

A function should do exactly what is suggested by its name, no more.

Two independent actions usually deserve two functions, even if they are usually called together (in that case we can make a 3rd function that calls those two).

A few examples of breaking this rule:

- `getAge` – would be bad if it shows an `alert` with the age (should only get).
- `createForm` – would be bad if it modifies the document, adding a form to it (should only create it and return).
- `checkPermission` – would be bad if it displays the `access granted/denied` message (should only perform the check and return the result).

These examples assume common meanings of prefixes. You and your team are free to agree on other meanings, but usually they're not much different. In any case, you should have a firm understanding of what a prefix means, what a prefixed function can and cannot do. All same-prefixed functions should obey the rules. And the team should share the knowledge.

Function Expressions

```
let sayHi = function() {
  alert( "Hello" );
};
```

Arrow functions

```
let func = (arg1, arg2, ..., argN) => expression;
```

```
let sum = (a, b) => a + b;
```

```
let sum = (a, b) => {  // the curly brace opens a multiline function
  let result = a + b;
  return result; // if we use curly braces, then we need an explicit "r
};
```

Callback functions

Any function that is passed as an argument to another function so that it can be executed in that other function is called as a callback function.

Recursion

There are two ways to implement it.

1. Iterative thinking: the `for` loop:

```
function pow(x, n) {
  let result = 1;

  // multiply result by x n times in the loop
  for (let i = 0; i < n; i++) {
    result *= x;
  }

  return result;
}

alert( pow(2, 3) ); // 8
```

2. Recursive thinking: simplify the task and call self:

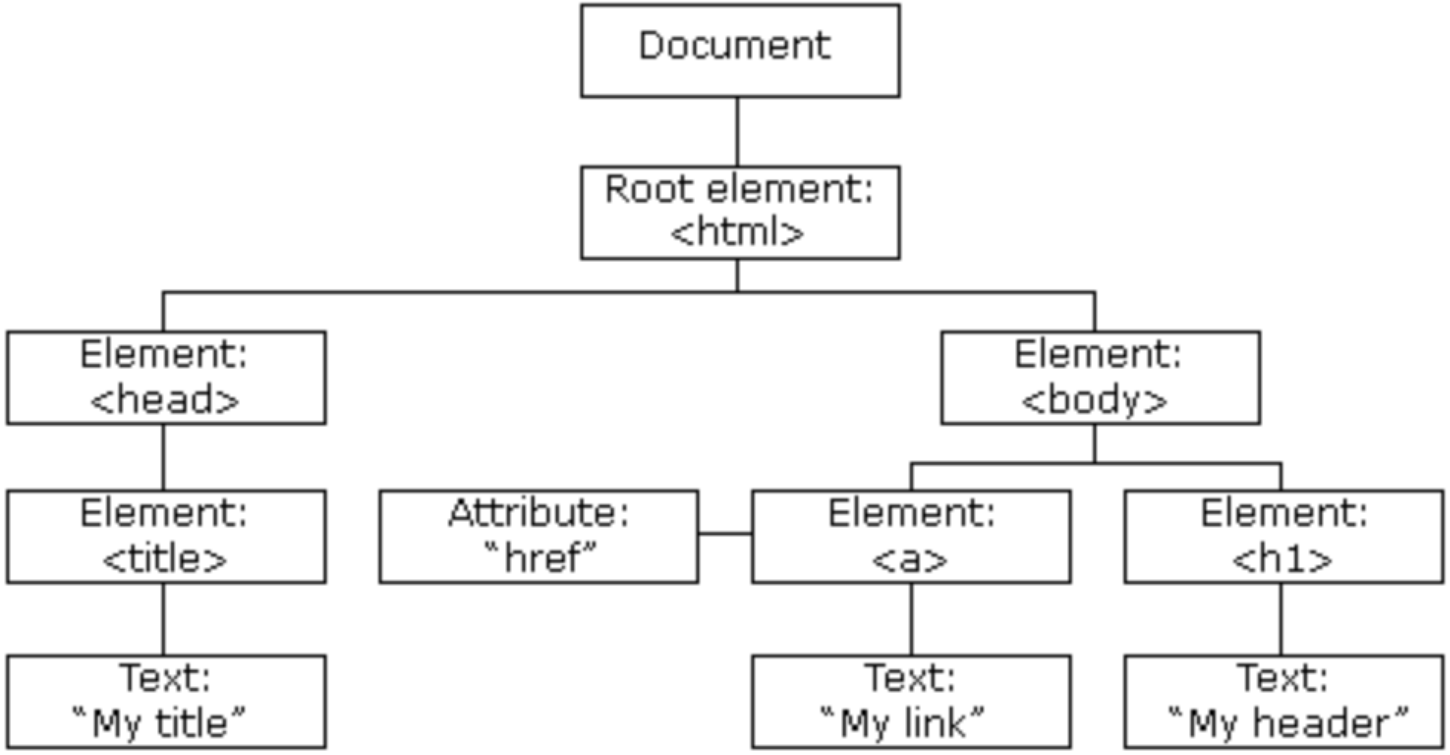
```
function pow(x, n) {
  if (n == 1) {
    return x;
  } else {
    return x * pow(x, n - 1);
  }
}

alert( pow(2, 3) ); // 8
```

When a function solves a task, in the process it can call many other functions. A partial case of this is when a function calls *itself*. That's called *recursion*.

Tasks

1. Create a function which calculate sum of 2 arguments and return it
2. Create a function which finds minimum value of 2 arguments and return it.
3. Create function which receive number and check if that number is prime or not.
4. Write a function which receive number and if number is bigger than 10 calls (cb1) and (cb2) otherwise
5. Create a functions which receive a number(string length) as argument and generates random string which received length. (Use Math.random)

JavaScript HTML DOM



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

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

```
const x = document.querySelectorAll("p.intro");
```

Changing HTML Elements

Property	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.style.*property = new style*	Change the style of an HTML element
Method	**Description**
element.setAttribute*(attribute, value)*	Change the attribute value 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(*new, old*)	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

Form Validating

```
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">
  Name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>
```

```
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

DOM Events

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

```
<h2 onclick="changeText(this)">Click on this text!</h2>

function changeText(id) {
  id.innerHTML = "Ooops!";
}
```

```
 document.getElementById("myBtn").onclick = displayDate;

document.getElementById("myBtn").addEventListener("click", displayDate);
```

DOM Events list

click	The event occurs when the user clicks on an element	MouseEvent
blur	The event occurs when an element loses focus	FocusEvent
change	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)	Event
copy	The event occurs when the user copies the content of an element	ClipboardEvent
cut	The event occurs when the user cuts the content of an element	ClipboardEvent
dblclick	The event occurs when the user double-clicks on an element	MouseEvent
submit	The event occurs when a form is submitted	Event

Tasks

1. **Create a form and validate inputs on submit**
2. **Implement Percentage Calculator**