JavaScript 1 - Module 1

Getting Started

Variables

Making Decisions

Loops

JavaScript

JavaScript (/ˈdʒɑːvəˌskrɪpt/), often abbreviated as JS, is a programming language that conforms to the ECMAScript specification.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web.

JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

Source: Wikipedia

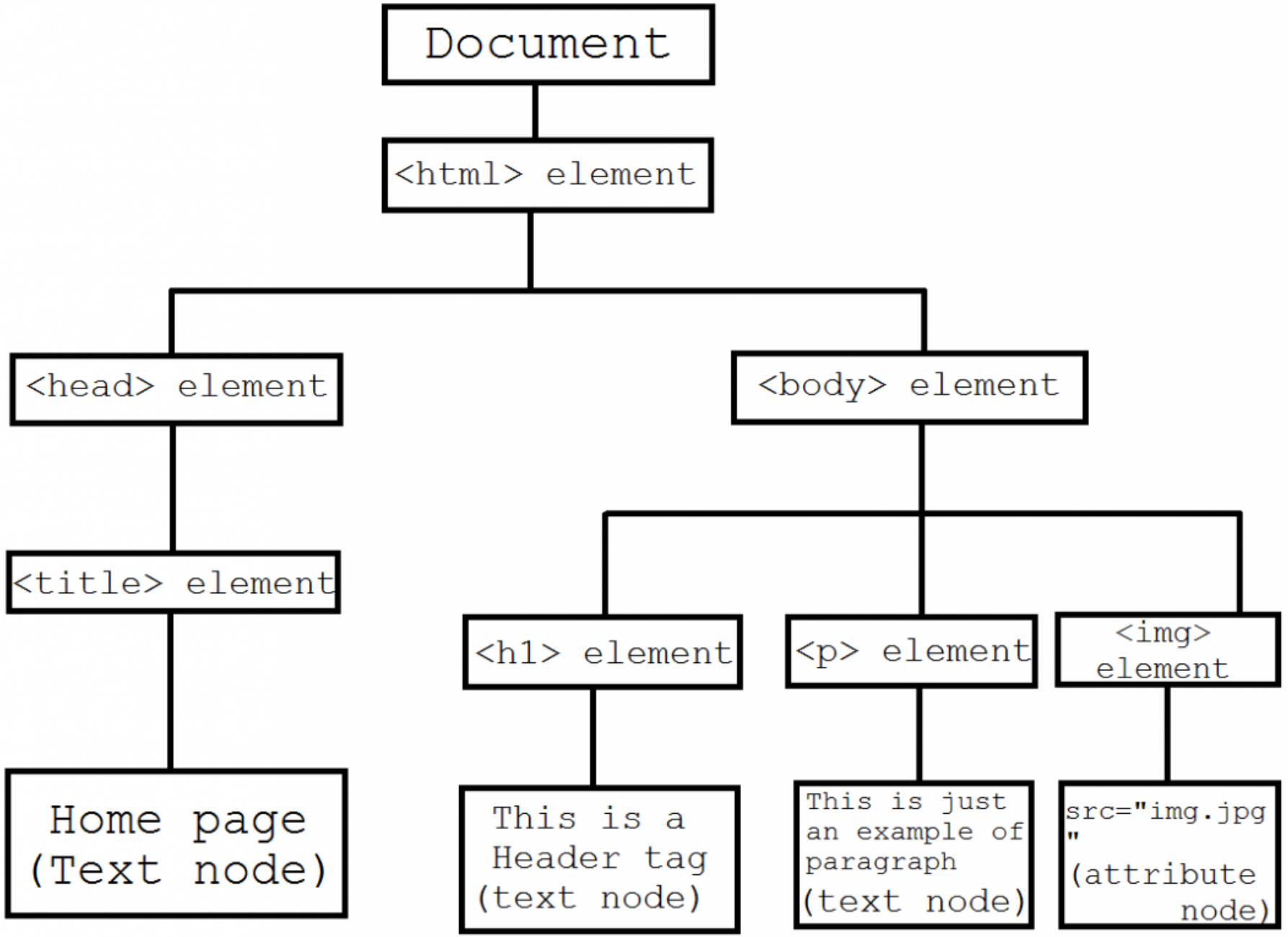
DOM, the Document Object Model

The Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a document on the web.

With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

When a web page is loaded, the browser creates a `Document Object Model` of the page, as a `tree` of Objects.

Each element in the DOM is also called a `node`.


Document
<html> element
<head> element
<body> element
<title> element
<h1> element
<p> element
<img> element
Home page (Text node)
This is a Header tag (text node)
This is just an example of paragraph (text node)
src="img.jpg" (attribute node)

Source

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

- JavaScript can change all the HTML `elements` on a page
- JavaScript can change all the HTML `attributes` on a page
- JavaScript can change all the CSS `styles` on a 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` on a page
- JavaScript can `create new` HTML `events` on a page

ES6 Harmony

ECMAScript 6, also known as ES6 and ECMAScript 2015, was the second major revision to JavaScript.

ES6 adds significant new syntax for writing complex applications (see next slide for extensive list of new features).

ES7 (2016) through ES14 (2023) has, incrementally, added new features, building on ES6.

New Features in ES6			
The `let` keyword	`String.startsWith()`	Set Objects	New Math Methods
The `const` keyword	`String.endsWith()`	Classes	New Number Properties
Arrow Functions	`Array.from()`	Promises	New Number Methods
The `...` Operator	`Array keys()`	Symbol	New Global Methods
For/of	`Array find()`	Default Parameters	Object entries
Map Objects	`Array findIndex()`	Function Rest Parameter	JavaScript Modules
`String.includes()`			

Source: New Features in ES6 (with examples)

Using Javascript

JavaScript can "display" data in different ways:

- Writing into an HTML element, using `innerHTML`.
- Writing into the HTML output using `document.write()`.
- Writing into an alert box, using `window.alert()`.
- Writing into the browser console, using `console.log()`.

> Warning: Using `document.write()` after an HTML document has finished loading, will delete all existing HTML

Using `innerHTML`

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 7 + 6;
</script>

</body>
</html>
```

Codepen

Using `document.write()`

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(7 + 6);
</script>

</body>
</html>
```

Codepen

Using `document.write()` after load

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<button type="button" onclick="document.write(7 + 6)">
    Click me!
</button>

</body>
</html>
```

> The document.write() method should only be used for testing.

Codepen

Using `window.alert()`

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<button type="button" onclick="window.alert(7 + 6)">
    Click me!
</button>

</body>
</html>
```

> Tip: You can skip the window keyword and just use `alert(7 + 6);`

Codepen

Using `console.log()`

```
<!DOCTYPE html>
<html>
<body>

Open Developer tools > Console to see output

<script>
console.log(7 + 6);
</script>

</body>
</html>
```

The console.log() method writes a message to the **console**.

The **console** is very useful for testing purposes.

Codepen

The Console

The Console is a part of the browser's web development tools, to view messages issued by JavaScript.

Web development tools allow you to test and debug the JavaScript code. Web development tools are often called devtools.

Modern web browsers such as Google Chrome, Firefox, Edge, Safari, and Opera provide the devtools as built-in features.

Generally, devtools allow you to work with a variety of web technologies such as HTML, CSS, DOM, and JavaScript.

Open the Console in some browsers:

Chrome: F12 or Ctrl + Shift + J (Cmd + Option + J on Mac)

or use the menu > View > Developer > JavaScript Console

Microsoft Edge: F12

or by clicking on the action button (top-left corner) > More tools > Developer Tools, then Console

Mozilla Firefox: Ctrl + Shift + K (or Cmd + Shift + K on a Mac, or F12)

or use the menu > Tools > Browser Tools > Web Developer Tools and choose Console

Safari: Cmd + option + C

or open it by accessing the Develop tab at the top and clicking on Show JavaScript Console.

NOTE: In Safari **Developer mode** must be activated in Safari preferences > Advanced (by checking Show Develop menu in the menu bar)

JavaScript Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable.

JavaScript comments can also be used to prevent execution, when testing alternative code. (See last line in example below.)

```
// This is a single line comment

/*
    This is a multi-line comment
    That can span over - well - multiple lines
*/

console.log("Hello"); // comment on this statement

// console.log("World");
```

Where To Put Your JavaScript

The `<script>` Tag

In HTML, JavaScript code is inserted between `<script>` and `</script>` tags.

```
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

> Old JavaScript examples may use a type attribute: `<script type="text/javascript">`.
> The type attribute is not required. JavaScript is the default scripting language in HTML.

JavaScript in `<head>` or `<body>`

You can place any number of scripts in an HTML document.

Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both.

JavaScript in `<head>`

In the example on the next slide, a JavaScript function is placed in the `<head>` section of an HTML page.

The function is invoked (called) when a button is clicked:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

JavaScript in `<body>`

In the next example, a JavaScript function is placed in the `<body>` section of an HTML page.

The function is invoked (called) when a button is clicked:

> Placing scripts at the bottom of the `<body>` element improves the display speed, because script interpretation slows down the display. Also, it ensures that the DOM elements are in place when the JavaScript is running.

```
<!DOCTYPE html>
<html>
<body>

<h2>Demo JavaScript in Body</h2>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

External JavaScript

Scripts can also be placed in external files:

External file: `myScript.js`

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension `.js`.

To use an external script, put the name of the script file in the src (source) attribute of a `<script>` tag:

```
<script src="myScript.js"></script>
```

You can place an external script reference in `<head>` or `<body>` as you like.

The script will behave as if it was located exactly where the `<script>` tag is located.

> External scripts cannot contain `<script>` tags.

> If you place the `<script>` tags in `<head>` consider using the `defer` attribute, to indicate to a browser that the script is meant to be executed after the document has been parsed, but before firing `DOMContentLoaded`.
> **Warning**: `defer` must not be used if the `src` attribute is absent (i.e. for inline scripts), in this case it would have no effect. The defer attribute also has no effect on module scripts — they defer by default.

External JavaScript Advantages

Placing scripts in external files has some advantages:

- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads

To add several script files to one page - use several script tags:

```
<script src="./js/myScript1.js"></script>
<script src="./js/myScript2.js"></script>
```

Todos

Github Classroom

JS1 Lesson 1.1 Getting Started

Mollify

Read Getting Started and do the Activity and Lesson Task

Extra Todos

Pick Chrome DevTools or Firefox Developer Tools and go through the Console tutorials:

* Chrome DevTools and Console Overview
* Firefox Developer Tools and Web Console

Extra Resources

JavaScript Tutorial

MDN Web Docs: JavaScript

- Learn web development > Getting started with the Web > JavaScript basics

Scratch

Scratch is the world's largest coding community for children and a coding language with a simple visual interface that allows young people to create digital stories, games, and animations.