JS CHEAT SHEET

2nd semester

Created by: Victor Schweitz

ARRAYS

Arrays

An array stores several values in one variable, which makes the code more manageable and easier to work with. For example, instead of creating a variable for each person, you can collect all family members in an array.

```
const arr = []
```

Syntax

```
const arr =
['element', 'element']
```

First, const is written followed by the variable name. Next comes an equals sign, followed by a set of square brackets indicating the array. Within these, the elements are separated by a comma.

OBJECTS

Objects

Objects group together information that belongs to the same thing, such as the gender, age, and name of a person. Instead of creating a variable for each piece of information, it is more manageable to collect them in one object.

```
const obj = {}
```

Syntax

```
const obj = {
   gender: ‘male’,
   age: 24,
   name: ‘Victor’,
}
```

Objects are always set equal to a variable. The syntax consists of const, variable name, an equals sign, and curly brackets, in which the key/value pairs are found. The key is followed by a colon and then the value. The key/value pairs are separated by commas.

FETCH

Fetch

Fetch is a built-in JavaScript function that retrieves data from other websites or databases via a URL. It makes it possible to update content dynamically without reloading the entire page.

```
fetch()
```

Syntax

```
fetch(‘url’)
```

First you write fetch followed by a set of parentheses. Inside these comes a set of quotation marks, in which the URL containing the fetch data is. There is also formatting, which can be found in the example linked on the last page.

QUERYSELECTORALL

.querySelectorAll()

.querySelectorAll() selects all elements that match the specified selector, while .querySelector() selects only the first one. .querySelectorAll() returns a NodeList that can be accessed like an array.

```
.querySelectorAll()
```

Syntax

```
const elements =
document.querySelectorAll
('.element')
```

First you write const, which is followed by the name of the variable. Then comes an equal sign followed by document.querySelectorAll(). Inside the parentheses is a set of quotes containing the selector.

FOREACH

.forEach()

.forEach() is a loop type that runs for each element of an array or NodeList returned by .querySelectorAll(). .forEach() can be used on both an array and a NodeList.

```
.forEach()
```

Syntax

```
const elements =
document.querySelectorAll
('.element')

elements.forEach(sayHello)

function sayHello(parameter) {
    // Code here
}
```

First, a variable is created, which is equal to an array or document.querySelectorAll(). Then the variable name is written followed by .forEach(), which parameter is the name of the function to be called. Finally, this function is declared.

TEMPLATE

Template

Template is an HTML element that acts as a placeholder for content generated by JavaScript. The content only becomes visible when it is added to the browser via methods such as .cloneNode(), .append(), .appendChild() and the .content attribute.

```
.cloneNode(),
.append(),
.appendChild(),
.content
```

Syntax

The syntax is fairly straightforward as it consists of an opening and closing element with content in - exactly like most other elements.

The generation of the content can be seen in the example on the last page.

```
<template>
  // Content here
</template>
```

URL PARAMETERS

URL parameters

URL parameters are used to send data between pages via the URL, creating a better flow on the website. Webshops use this technique to send items from the overview page to the basket and on to the payment page.

```
URLSearchParams()
```

Syntax

First comes the base URL, which is the one that ends with ".html". Next comes a question mark indicating that the first parameter is coming, then the name of the first parameter, which is followed by an equal sign and the value of the parameter.

```
base-url.html?parameter=value
```

ATTRIBUTES

Attributes

Attributes are used to make an element more specific. This can be with attributes like class or id. These can also be set dynamically via JavaScript with .setAttribute() and retrieved with .getAttribute().

```
<div class=”square” id=”square”>
```

Syntax

```
// Getting the element
const div =
document.querySelector(‘div’)

// Giving it the “square” class
div.setAttribute(‘class’, ‘square’)

// Getting the class attribute
div.getAttribute(‘class’)
```

Both .setAttribute() and .getAttribute() are applied to the variable. .setAttribute() takes two parameters: the attribute and its value - both in quotes and separated by commas. .getAttribute() takes only one quoted parameter, which is the attribute set in .setAttribute().

EXAMPLES