JavaScript

Anton Bäckström

Why use JavaScript

- There are 3 core technologies of any website

1. HTML
   - Decides the content of the page
2. CSS
   - Decides the look of the page
3. JavaScript
   - Controls the behaviour of the page

Why use JavaScript

- Without JavaScript you can only have static pages
- JavaScript can modify both the HTML and CSS in realtime

How to use JavaScript

- There are 2 ways to include JavaScript in your page

1. Inline code

<script> JavaScript code </script>

2. External code

<script src=”external.js”></script>

Dynamic types

- JavaScript is a dynamically typed language, so you create variables using
  - var
  - const
  - let

- Example

```
var myList = [1, 3, 3, 7];
var myInteger = 1337;
var myString = '1337';
```

Dynamic types

- You may change types on the fly

```
var myList = [1, 3, 3, 7];      // type = array
myList = 1337;                  // type = number
myList = '1337';                // type = string
myList = {a: 1337};             // type = object
```

Dynamic types

- JavaScript is sometimes too easy on types

```
function foo (input) {
    if (input == 1)
            return input + 1;
    return 0;
}

foo(1);          // 2
foo('1');        // 11
foo(true);       // 2
```

Dynamic types

- If you wish to ensure the data-type is correct

```
function foo (input) {
    if (input ===  1)
            return input + 1;
    return 0;
}

foo(1);             // 2
foo('1');           // 0
foo(true);          // 0
```

Dynamic types - Truthy / Falsey

Value	Truth-value
true	true
false	false
''	false
'hello'	true
'true'	true
'false'	true
-1	true
0	false
1	true
2	true
[]	true
[false]	true
{}	true
undefined	false
null	false

Debugging

- Look at current value in console
- alert()
  - Halts the current thread to show a popup
- console.log()
  - Prints message to the console in the developer tools
- innerHTML & document.write()
  - Enter the value into the HTML to display on screen

Functions

- There are 2 types of functions
  - Normal functions
    function foo(param) {}
    const foo = funtion(param) {};
  - Arrow functions, do not bind *this* (new in ES6)
    const foo = (param) => {};
    const foo = param => {};
    const foo = param => output;

Functions

- Functions are treated as any other variable, which means you can pass them as arguments

```
function add(a, b) { return a + b; }
function sub(a, b) { return a - b; }
function mult(a, b) { return a * b; }
function apply(a, b, functions) {
    for (const f of functions)
        a = f(a, b);
    return a;
}

apply(1, 2, [add, sub, mult]);          //  1 + 2 – 2 * 2 = 2
```

Functions

- Be wary of *callback hell*

```
function handler(value, success, fail) {
    if (value) { success(); }
    else { fail(); }
}

const value = false;
const res = handler( value,
                     () => 'Success !',
                     () => 'Fail !'       );

// res = undefined
```

Objects

- Objects store everything using key-value pairs

```
var myObject = {
    str: '1337',
    list: [1, 3, 3, 7],
    num: 1337,
    func1: param => param + this.num,
    func2: function(param) { return  param + this.num; },
}

myObject.str;          // '1337'
myObject.func1(1);     // NaN
myObject.func2(1);     // '1338'
```

Useful built-in Objects

- The most important object might very well be the *document*-object
- document is used to interact with the HTML

```
document.getElementsByTagName('body')[0].innerHTML = 'Hello World';
document.getElementById('subscribe').checked = true;
document.getElementsByClassName('hidden')[0].style.display = 'none';
```

- We can also access the local-storage or cookies to store permanent information

```
const highscores = localStorage.highscores;
const cookie = document.cookie;
```

Events

- Lets you perform actions when something happens
  - onload
  - onlick
  - onmouseover
  - onfocus
  - etc…

Useful array-functions

// Only keep elements that pass the function
Array.filter();

// Perform the function on each entry to generate a comulative result
Array.reduce();

// Perform the function on each entry, return the results as an array
Array.map();

// Sort the array in increasing order
Array.sort();

// Reverse the order of the list
Array.reverse();

// Return a sub-section or the array
Array.splice();

// Return the concatenation of 2 lists
Array.concat();

// Convert the list to a string by separating elements with the given value
Array.join();