WEB DESIGN DECAL

LECTURE 8

Introduction to Javascript

Basic programming skills

Eric Liang

JS

5

HTML

5

CSS

3

Motivation for JavaScript

Where have we been?

- Learned how HTML organizes a web page, and not much more
- To style those HTML elements, we select for them with good ol' CSS. Now they look good!
- CSS also powers the ability to style elements when you hover over them. But how can we take this further?

Motivation for JavaScript

What do we still want to do?

Make things react to:

- click events
- scrolling
- keyboard presses, etc.

Plugins for sliders, search, galleries, everything!
Do some form validation logic
Animations!

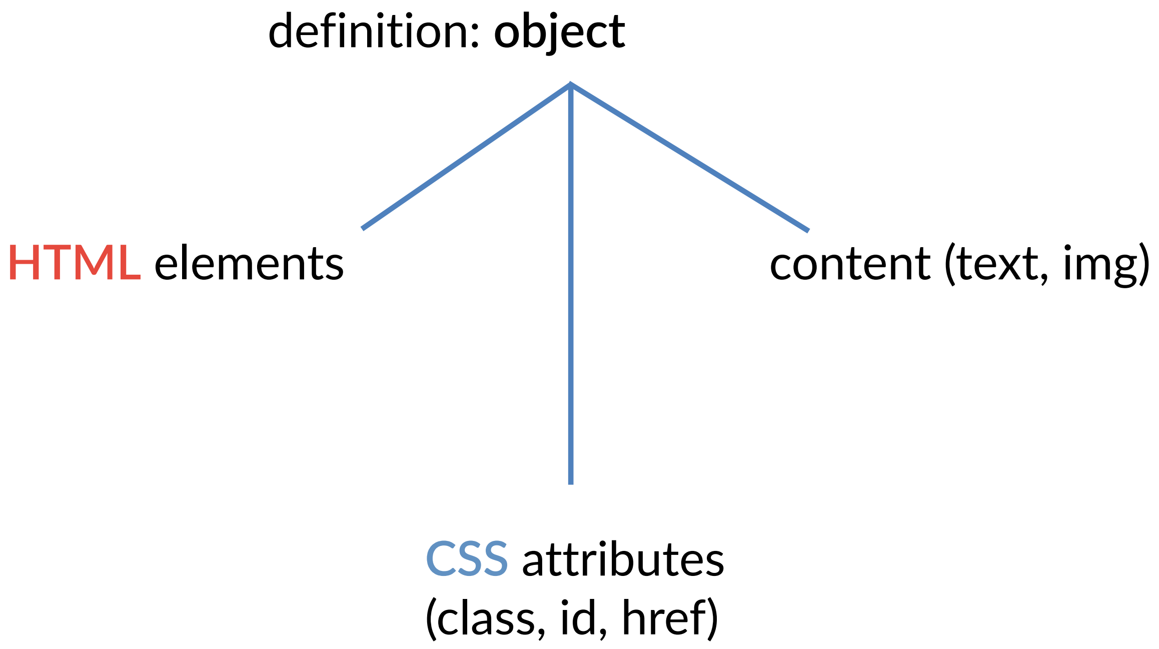
What is JavaScript?

Javascript is <u>not</u> Java!

A dynamic programming language that can be applied to an HTML document to create dynamic interactivity on websites. Used to manipulate the DOM.

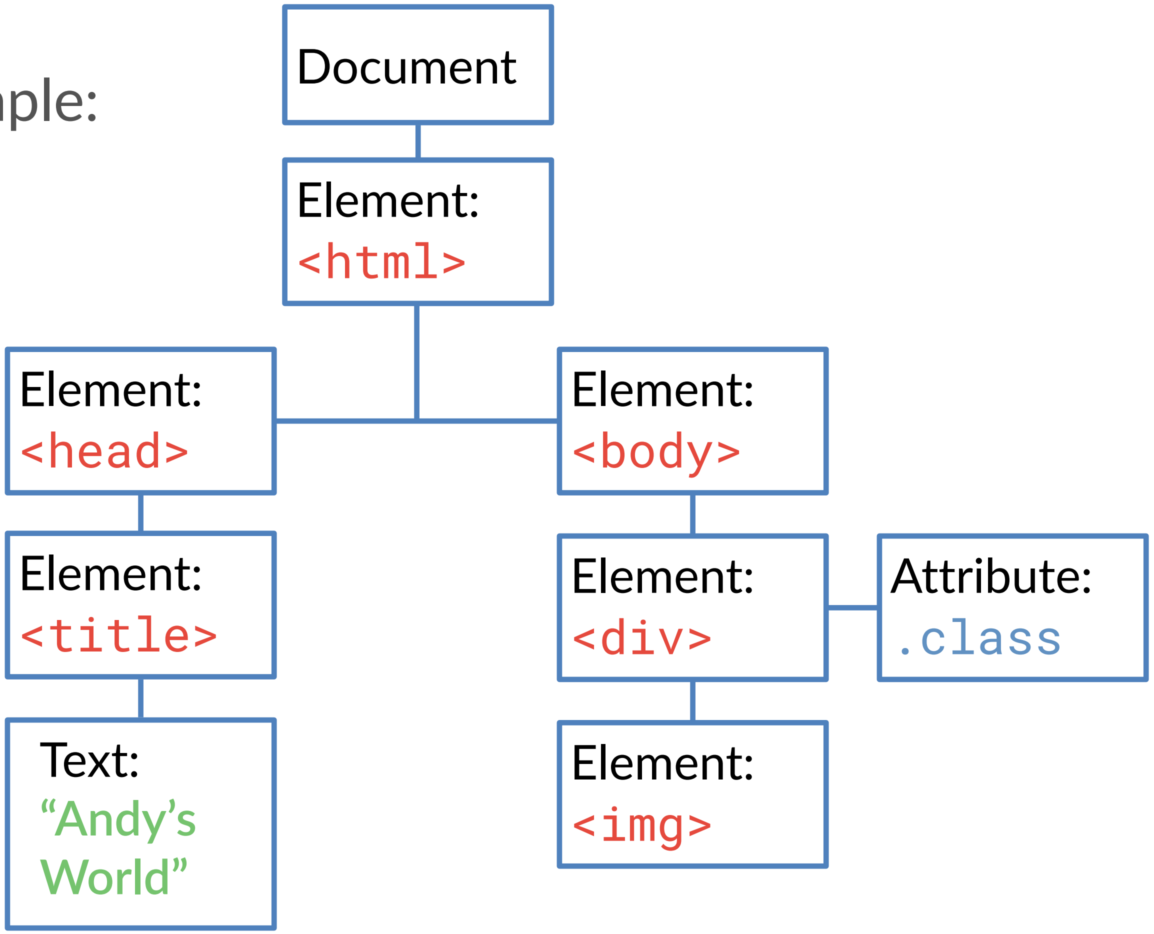
What's the DOM?

Stands for Document Object Model.
It's the tree of objects that JavaScript sees and controls.



What's the DOM look like?

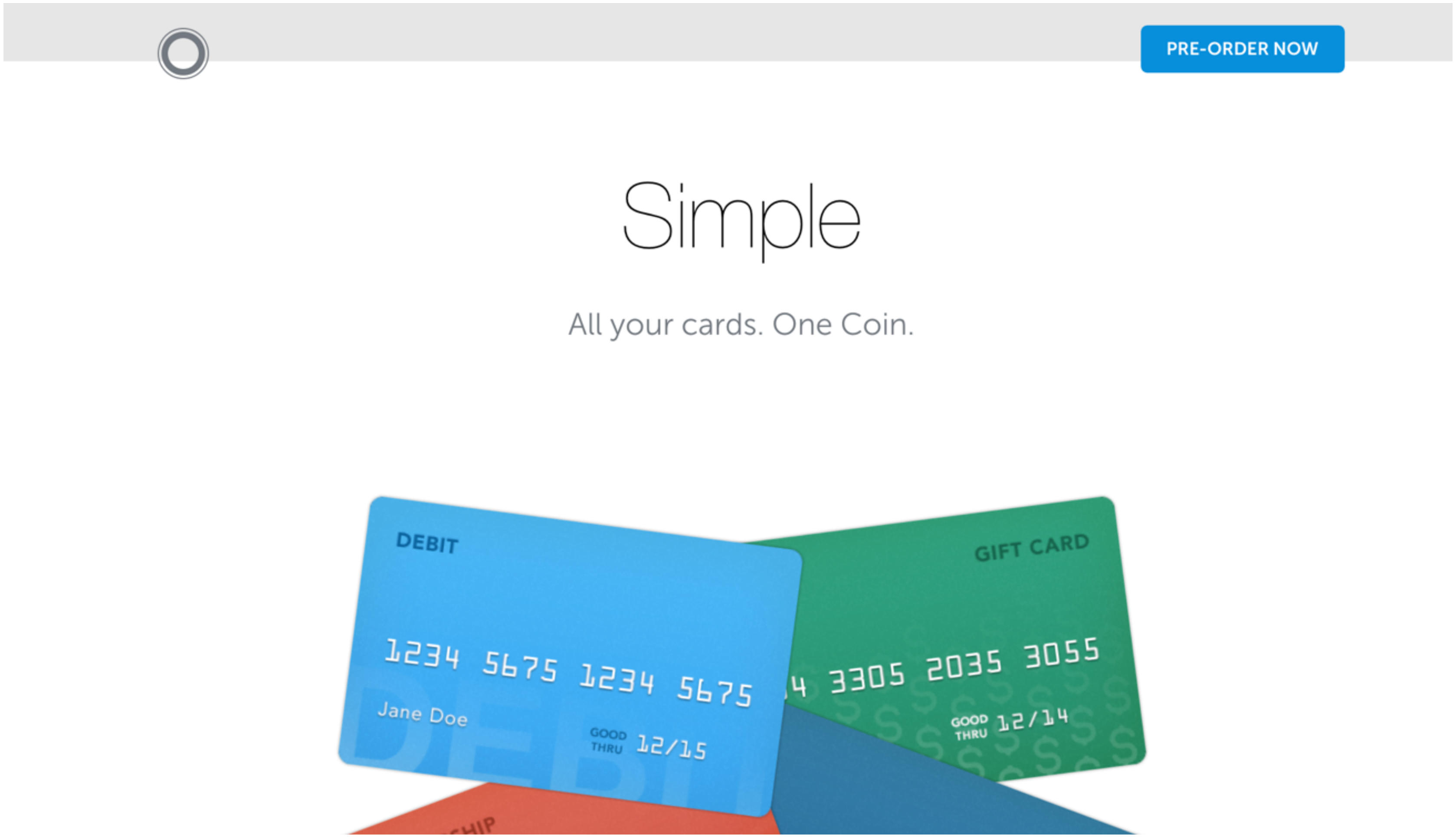
Here is a DOM example:



JavaScript with the DOM can

- change all HTML elements
- change any HTML attributes
- can add or remove HTML elements & attributes
- can change any of the the CSS styles on page
- reacts to all page events

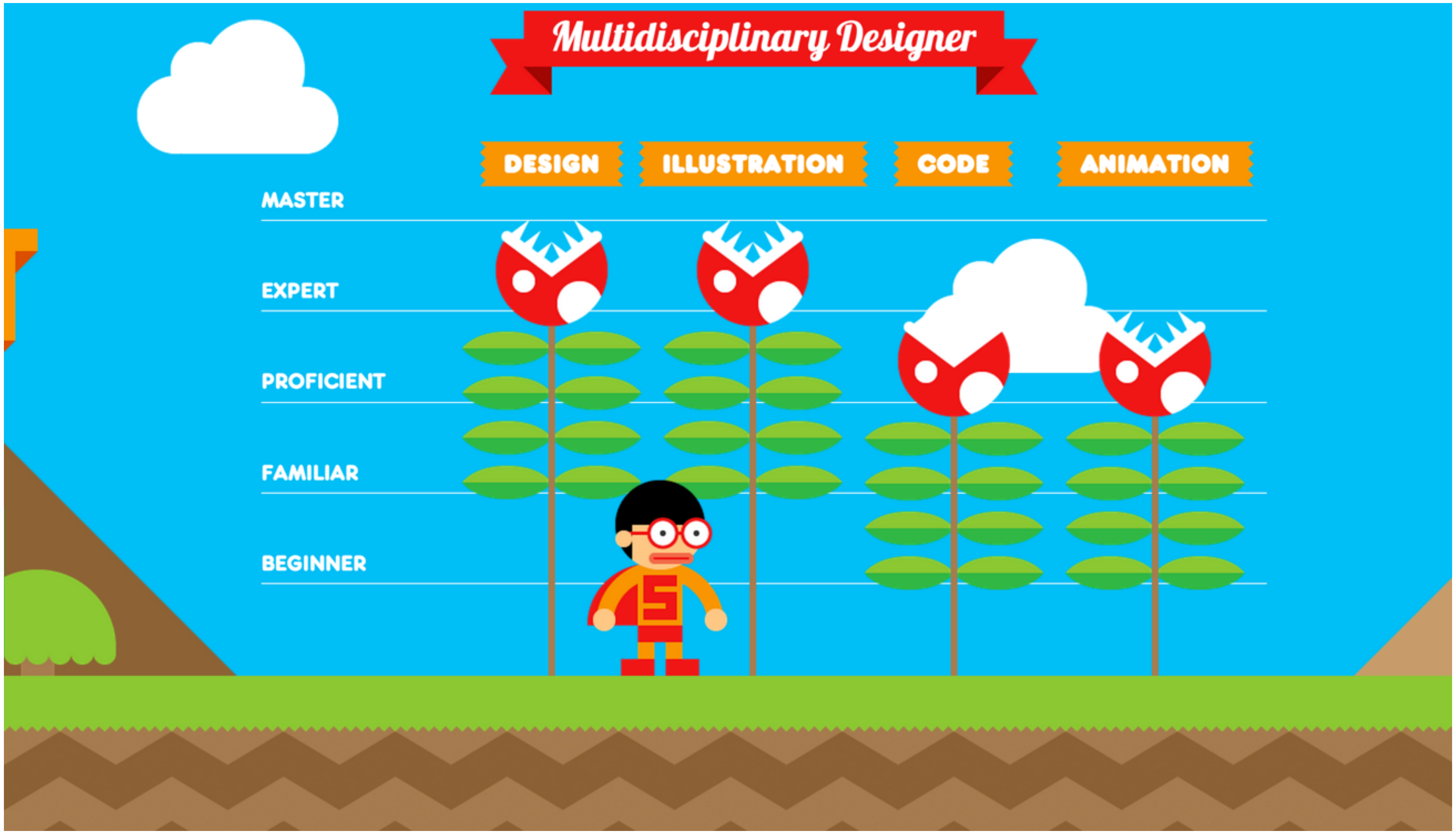
Everything above is referred to as DOM manipulation



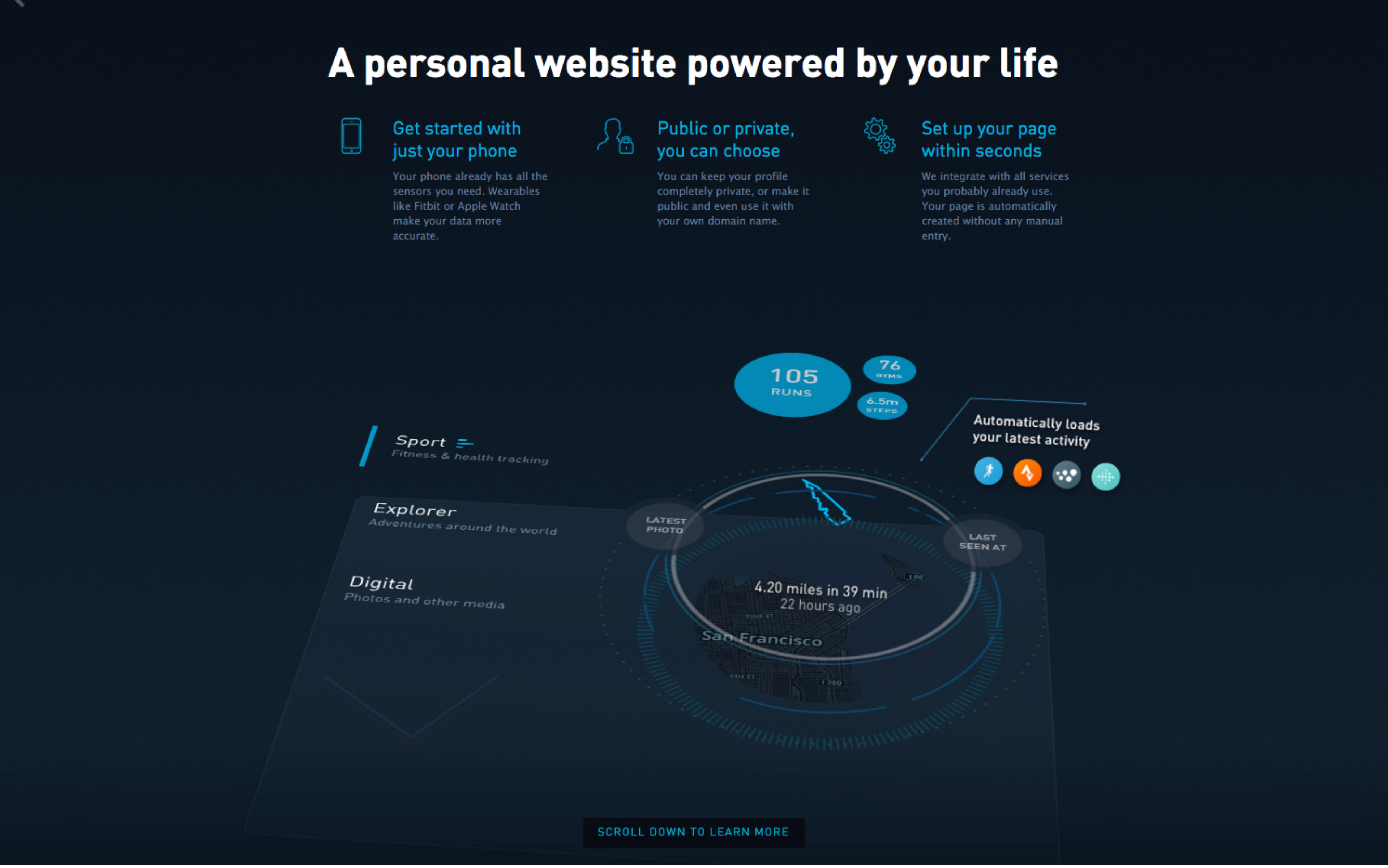
Simple

All your cards. One Coin.

Page animations - Coin



Interactive Resume - [Robby Leonardi]


A personal website powered by your life

Get started with just your phone

Your phone already has all the sensors you need. Wearables like Fitbit or Apple Watch make your data more accurate.

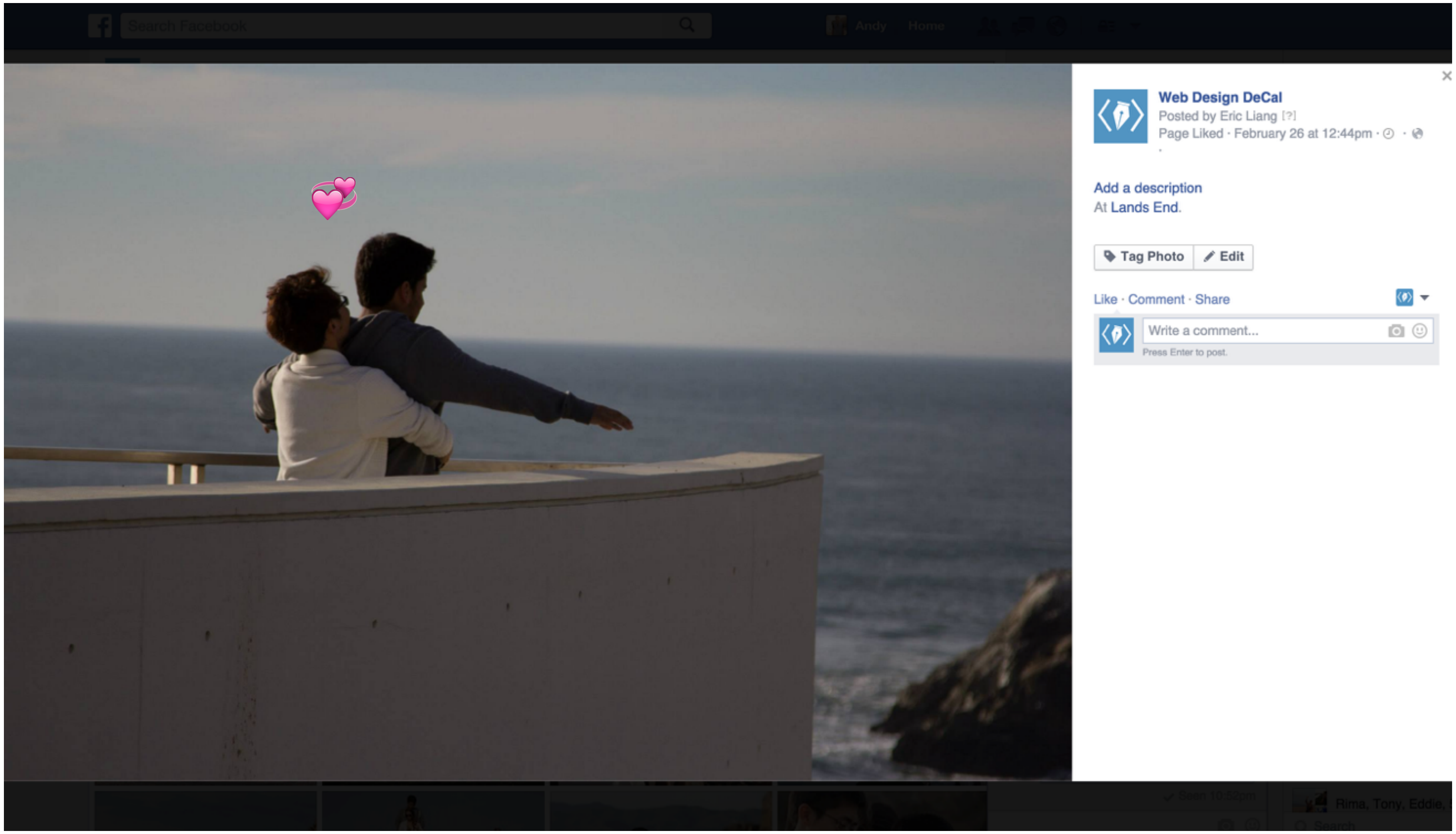
Public or private, you can choose

You can keep your profile completely private, or make it public and even use it with your own domain name.

Set up your page within seconds

We integrate with all services you probably already use. Your page is automatically created without any manual entry.

More animations - Gyroscope



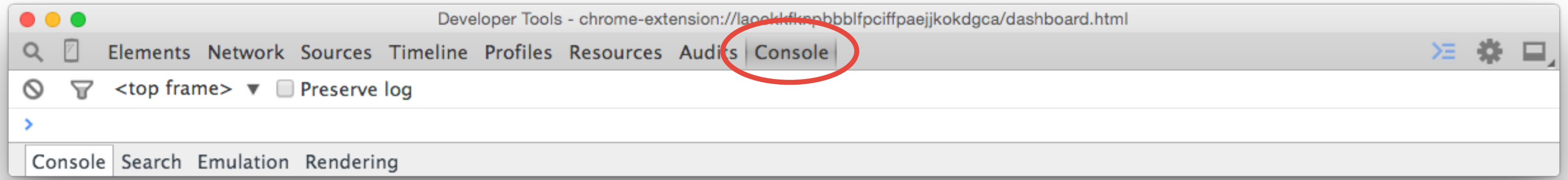
Modals - Facebook

JavaScript with the DOM

1. Link your JS file (like you do with CSS)

```
<script type="text/javascript" src="assets/js/
script.js"></script>
```

2. Add JavaScript on the fly in the console
Right click -> Inspect Element -> Console



Demo:

“Hacking” Google

Comments

A way to make comments in the code without worrying about the code being altered

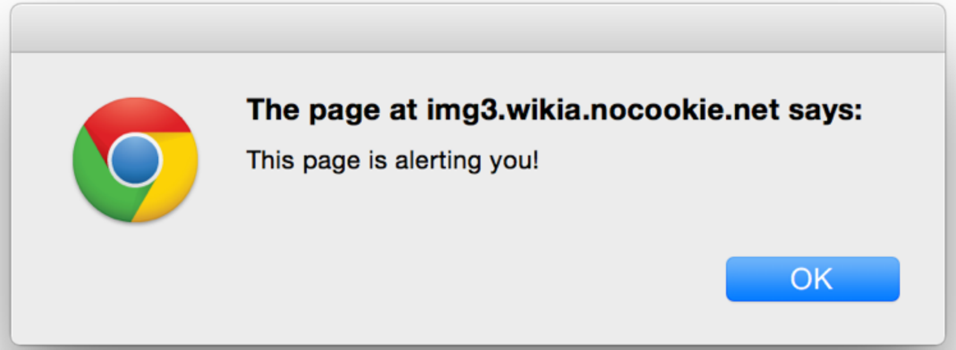
Use these “//” in front of comments

```
// This won’t be rendered on the page. Yay!
```

Alerts

A kind of popup box with the specified message.
Code:

```
alert("this page is alerting you!");
```



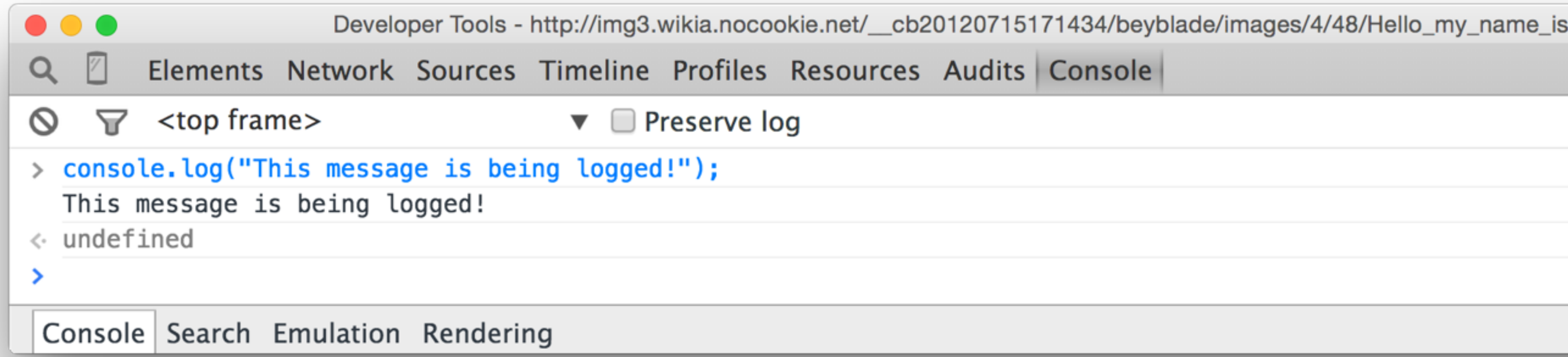
The page at img3.wikia.nocookie.net says:

This page is alerting you!

Logging (Printing)

A way to print things in the console, for testing purposes

```
console.log("This message is being logged!");
```


Developer Tools - http://img3.wikia.nocookie.net/__cb20120715171434/beyblade/images/4/48/Hello_my_name_is
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
Console Search Emulation Rendering

```
> console.log("This message is being logged!");
  This message is being logged!
<· undefined
>
```

Now to the fundamentals of programming.

Variables

You can think of variables as a label
Acting on the variable acts on what it represents

Variables

You can think of variables as a label
Acting on the variable acts on what it represents

```
var age = 19;
var name = "Canada He";
var allAboutThatBass = true;
```

Variables

Note that there’s a key difference between the following:

```
var x = 1;
x + 1;
// the value of x is still 1

var x = 1;
x = x + 1;
// the value of x has changed to 2
```

Types

Notice the 3 main types of values we've seen:

- `19` (a number)
- `“Canada He”` (a string, or words between quotes)
- `true` (a boolean, or `true`/`false` value)

Plenty of other types exist

Distinguishing types is important! To clarify:

- `“19”` (is a string)
- `False` (is not a boolean, must be all-lowercase)

Types - String

Strings are a kind of data type consisting of a sequence of characters between quotes.

To conjoin two or more strings, just use a + operator.

For example, in the console, try:

```
    "Hello," + " " + "world!"
```

Types - Boolean

There are booleans (true/false), and there are boolean expressions (expressions that are either true or false)

So whereas `true` is a boolean value,
`powerLevel > 9000;` is a boolean expression.

The var `powerLevel` could be a number greater than 9000 (so the expression is true) or less than/equal to 9000 (expression is false).

Types - Boolean

Here are some classic boolean operators for your convenience:

```
x == y   // "is x equal to y?"
x < y    // "is x less than y?"
x > y    // "is x greater than y?"
x <= y   // "is x less than or equal to to y?"
x >= y   // "is x greater than or equal to to y?"
!x       // “the opposite of x"
```

Where x and y are any boolean expressions.

Types - Boolean

Here are some classic boolean operators for your convenience:

```
x && y  // "are both x and y true?" (logical AND)
x || y  // "is either x or y true?" (logical OR)
x || !y // "is either x true or y false"
```

Where x and y are any boolean expressions.

Types - Boolean

Something to distinguish:

`==` vs. `=`

The one on the left is a boolean operator:
“Is what’s on the left the same as what’s on the right?”
(`true`/`false`)

The one on the right is an assignment operator:
“Assign what’s on the right to the variable on the left”

Conditionals

Code structures that allow you to do different things depending on another expression, a.k.a they test boolean expressions

```
if (day == "Wednesday") {
  var wearingPink = true;
} else {
  var wearingPink = false;
}
```

Conditionals

Code structures that allow you to do different things depending on another expression, a.k.a they test boolean expressions

```
if (YOUR BOOLEAN EXPRESSION HERE) {
  code that executes when true..
} else {
  code that executes when false..
}
```

Conditionals

There’s also an else if, for when you’ve got > 1 condition

```
if (product == "iphone") {
  console.log("It's an iPhone!");
} else if (product == "tablet") {
  console.log("It's an iPad!");
} else {
  console.log("It's a Mac!");
}
```

Conditionals

An exercise:

```
if ((x > 10) || (y <= -2)) {
  console.log("Victory!");
} else {
  console.log("Defeat :(");
}
```

What gets logged when `x = 11` and `y = 8`?
What gets logged when `x = 10` and `y = 8`?

Conditionals

```
x = 11;
y = 8;

    if (true “or” false)

if ((x > 10) || (y <= -2)) {
  console.log("Victory!");
} else {
  console.log("Defeat :(");
}
```

```
Victory!
```

Conditionals

```
x = 10;
y = 8;

   if (false “or” false)

if ((x > 10) || (y <= -2)) {
  console.log("Victory!");
} else {
  console.log("Defeat :(");
}
```

```
Defeat :(
```

Conditionals

Check if a number is even!

`x % y` finds the remainder when dividing x / y

```
7 % 2 == 1
4 % 2 == 0
```

```
if (x % 2 == 0) {
  // x is even!
} else {
  // x is odd!
}
```

Conditionals - Real-world use

Let’s say we have an application with users.

If a user is logged in, show their profile
If a user isn’t log in, show the homepage and Sign Up page

That’s a conditional, we have to use!

```
if (loggedIn) {
  // show profile
} else {
  // homepage
}
```

Loops

When you want to repeat the same code, some number of times. Happens often!

```
for (var i = 0; i < 10; i = i + 1) {
  console.log("This is what i is:" + i);
}
```

Loops

$$\texttt{for (}\underbrace{\texttt{var i = 0}}_{\text{“Let i start at 0”}}\texttt{; }\underbrace{\texttt{i < 10}}_{\text{“While i is less than 10”}}\texttt{; }\underbrace{\texttt{i = i + 1}}_{\text{“Let’s increase i by 1 each time”}}\texttt{)}$$

Functions

Definition: A set of code that performs a specific task.

In math, a function could be:

$f(x) = 3x + 10$

In JavaScript, the equivalent function would be:

```
function f(x) {
  return 3 * x + 10;
}
```

Functions

What does `return` mean? Indicates that “this is my output” for the function. Should be no code after the line with `return`

A function has 0 or more inputs (a.k.a. arguments) and returns at least one output

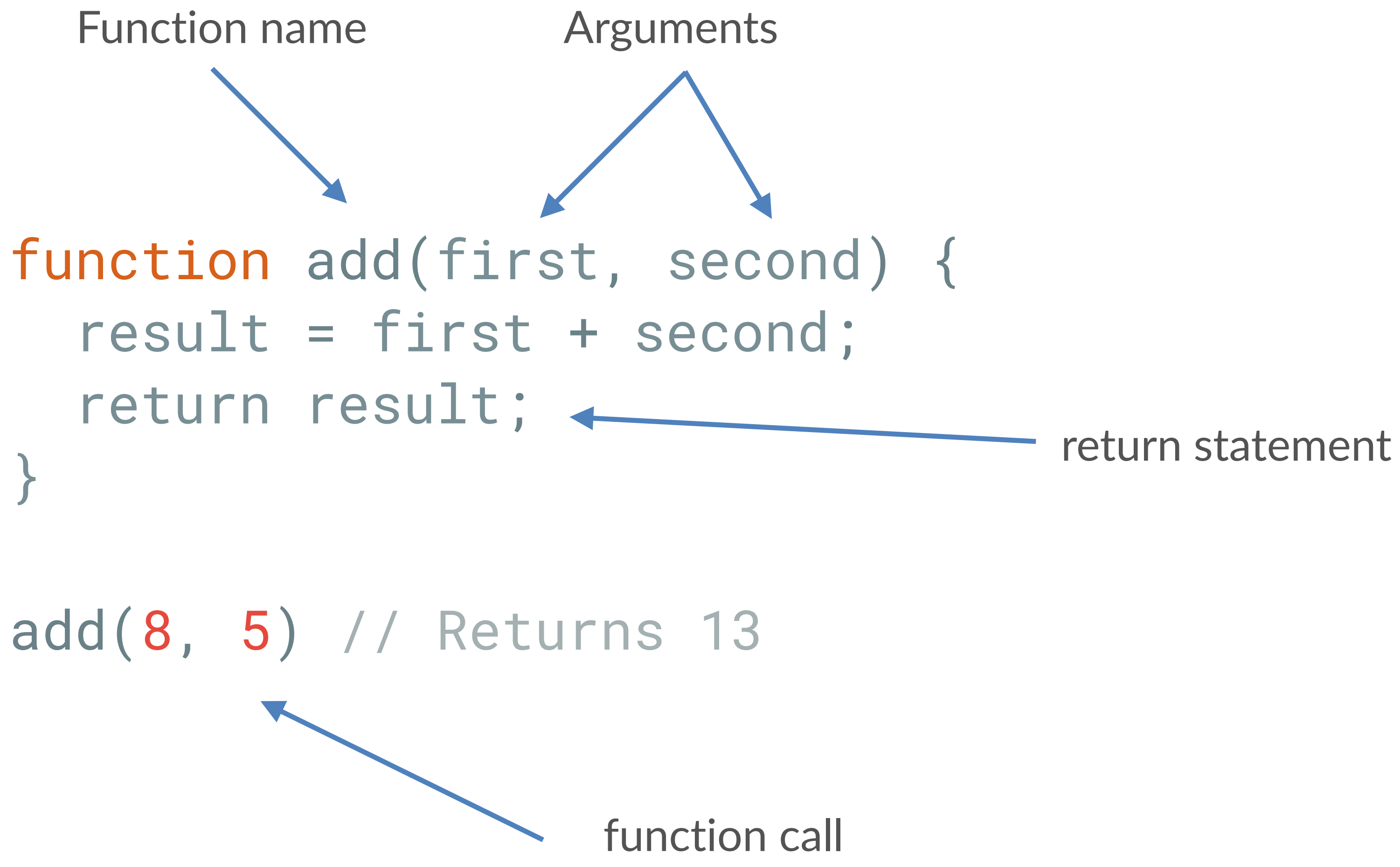
Functions - Anatomy of a Function

Function name   Arguments

```
function add(first, second) {
  result = first + second;
  return result;
}

add(8, 5) // Returns 13
```

Functions - Anatomy of a Function



```
function add(first, second) {
  result = first + second;
  return result;
}

add(8, 5) // Returns 13
```

Functions - Common Mistake

```
function foo(bar) {
  bar = 1;
  return bar;
}

foo(5)?
```

Functions - Common Mistake

```
function foo(bar) {
  bar = 1;
  return bar;
}

foo(5)?
```

The value of `bar` is overwritten, so it always returns 1

DOM manipulation

There's a series of functions that let you get and set DOM elements:

```
document.getElementById();
document.getElementByClassName();
document.getElementByTagName();
...
```

But thankfully, you won't have to memorize and type these long functions. Because we have jQuery.

Summary

Syntax = a `var` can be a number, string, boolean, etc.

Conditionals = `(if ... then ...)`

Functions = Code that does a specific task

JavaScript can manipulate the DOM, easiest done with:

jQuery

To be continued.