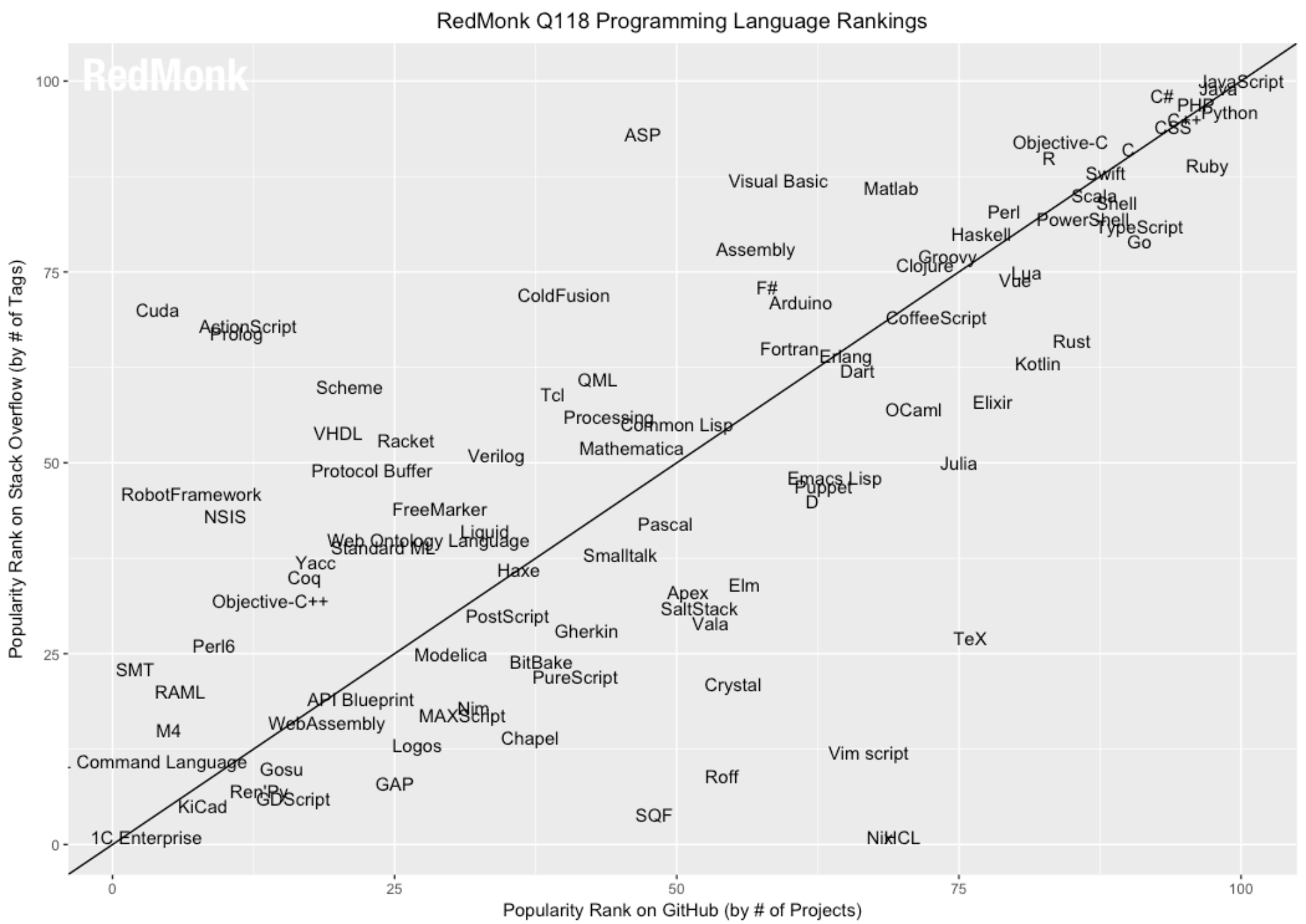


http://redmonk.com/sogrady/2018/03/07/language-rankings-1-18/

How to Open Your Browser’s JavaScript Console

Chrome:
Windows: Ctrl + Shift + **J**
Mac: Cmd + Opt + **J**

Firefox:
Windows: Ctrl + Shift + **K**
Mac: Cmd + Opt + **K**

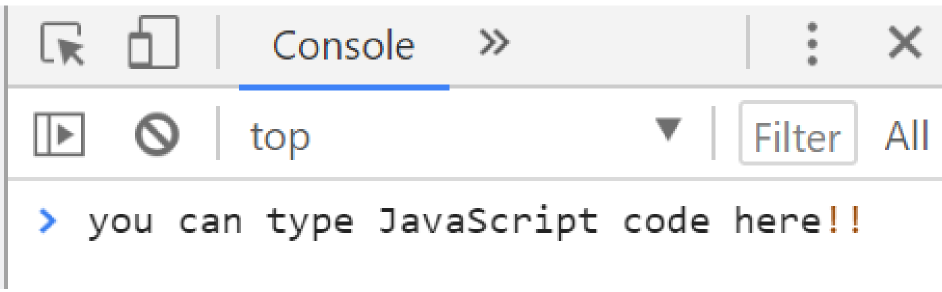
Safari / *Mac:* Cmd + Opt + **C**

For Safari: *enable web inspector first!*

1. Choose Safari > Preferences, and click Advanced.

2. At the bottom of the pane, select the “Show Develop menu in menu bar” checkbox.

3. Choose Develop > Show Web Inspector.



How to Open Your Browser’s JavaScript Console

Chrome:
Windows: Ctrl + Shift + **J**
Mac: Cmd + Opt + **J**

Firefox:
Windows: Ctrl + Shift + **K**
Mac: Cmd + Opt + **K**

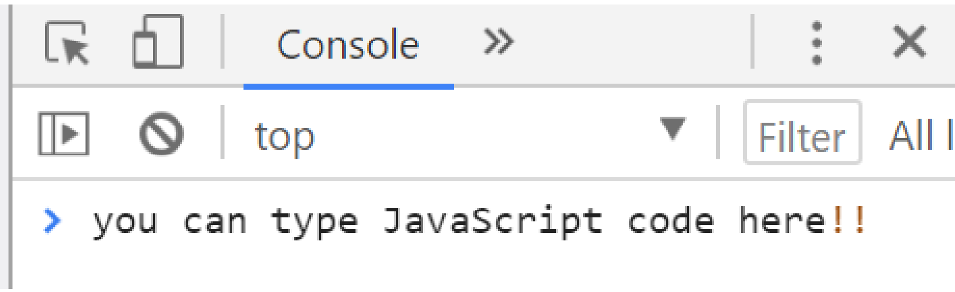
Safari / *Mac:* Cmd + Opt + **C**

For Safari: *enable web inspector first!*

1. Choose Safari > Preferences, and click Advanced.

2. At the bottom of the pane, select the “Show Develop menu in menu bar” checkbox.

3. Choose Develop > Show Web Inspector.



How to Open Your Browser’s JavaScript Console

Chrome:
Windows: Ctrl + Shift + **J**
Mac: Cmd + Opt + **J**

Firefox:
Windows: Ctrl + Shift + **K**
Mac: Cmd + Opt + **K**

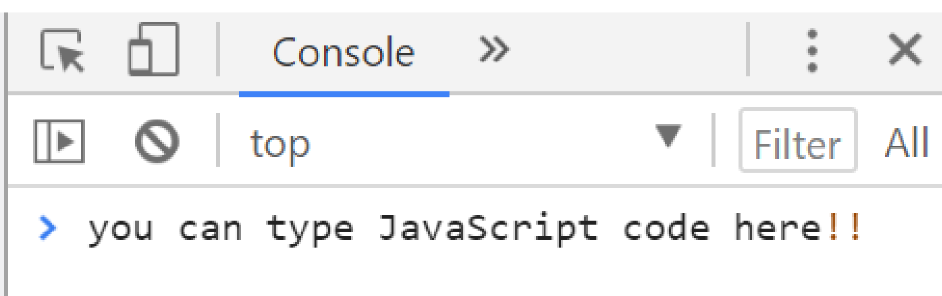
Safari / *Mac:* Cmd + Opt + **C**

For Safari: *enable web inspector first!*

1. Choose Safari > Preferences, and click Advanced.

2. At the bottom of the pane, select the “Show Develop menu in menu bar” checkbox.

3. Choose Develop > Show Web Inspector.



How to Open Your Browser’s JavaScript Console

Chrome:
Windows: Ctrl + Shift + **J**
Mac: Cmd + Opt + **J**

Firefox:
Windows: Ctrl + Shift + **K**
Mac: Cmd + Opt + **K**

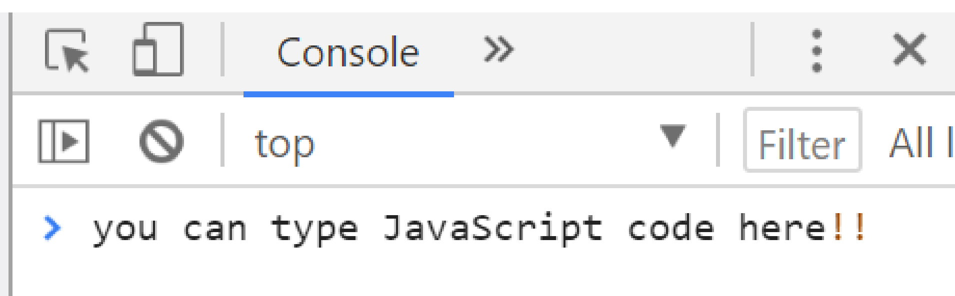
Safari / *Mac:* Cmd + Opt + **C**

For Safari: *enable web inspector first!*

1. Choose Safari > Preferences, and click Advanced.

2. At the bottom of the pane, select the “Show Develop menu in menu bar” checkbox.

3. Choose Develop > Show Web Inspector.



Memory

Name	Value

You can think of the computer's memory like a spreadsheet with two columns: name and value.

Memory

Name	Value

You can think of the computer's memory like a spreadsheet with two columns: name and value.

Memory

Name	Value
x	undefined

The **let** keyword creates a new empty slot, like a new "row" in the computer's memory.

```
let x;
```

Memory

Name	Value
x	undefined

The **let** keyword creates a new empty slot, like a new "row" in the computer's memory.

```
let x;
```

Memory

Name	Value
x	3

The **=** is called the "assignment operator". It assigns the value on the right into the slot for the variable name on the left.

```
x = 3;
```

Memory

Name	Value
x	3

The **=** is called the "assignment operator". It assigns the value on the right into the slot for the variable name on the left.

```
x = 3;
```

Memory

Name	Value
x	3
greeting	"hi"

You can also create a variable and *assign* a value to it all at once.

```
let greeting = "hi";
```

Memory

Name	Value
x	3
greeting	"hi"

You can also create a variable and *assign* a value to it all at once.

```
let greeting = "hi";
```

Memory

Name	Value

Challenge:
What will the memory look like after running the code below?

```
let someNum = 42;
let zipCode = "90017";
someNum = 90000 + 17;
let anotherVariable;
```

Memory

Name	Value

Challenge:
What will the memory look like after running the code below?

```
let someNum = 42;
let zipCode = "90017";
someNum = 90000 + 17;
let anotherVariable;
```

How to Link JavaScript Files to Your Web Page

index.html

```
<!DOCTYPE html>
<html>
 <head>
  <title>Hello!</title>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet" href="styles.css">

  <script src="script.js" defer></script>

 </head>
 <body>

  <h1>Welcome to My Awesome Web Page</h1>

  <p>Here's some text inside a paragraph.</p>

 </body>
</html>
```

script.js

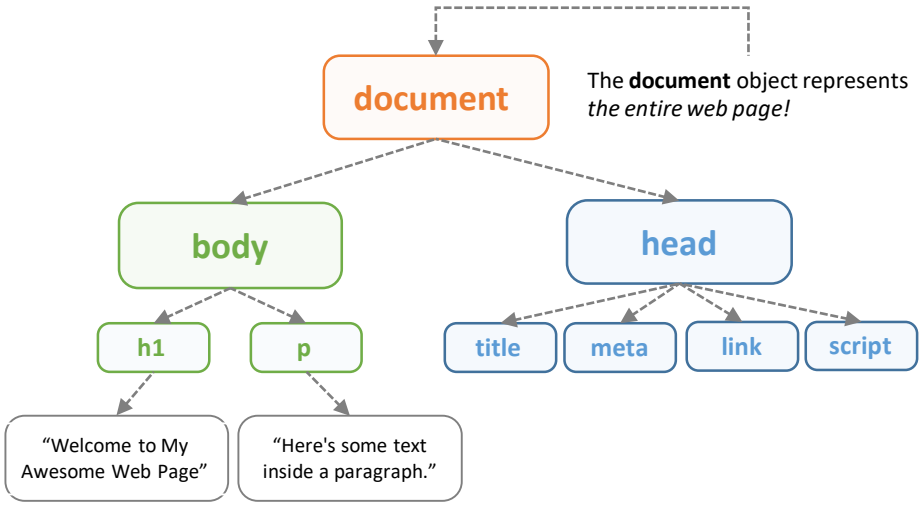
```
// This line (which starts with two forward slashes) is a comment!
// Comments are ignored by the computer.
// They're great for writing notes to yourself, so use them ALL the time!

console.log("Hello, world! This message comes from script.js");
```

1. The user navigates to the web page (something like http://example.com)
2. Their web browser downloads the HTML file (index.html is the default home page)
3. The <script> tag(s) in the HTML file tell the web browser which JavaScript file(s) to download next.
   ** The **defer** keyword inside the <script> tag tells the browser to WAIT until the HTML has finished downloading, and THEN download the JavaScript file afterwards.
4. The browser downloads the specified JavaScript file(s) and executes their code in the order that they were downloaded. (And order *does* matter here!)
5. The JavaScript code can then interact with the web browser, reacting to the user's actions and potentially modifying the HTML or CSS of the current web page.

The Document Object Model (DOM)

The Document Object Model (or **DOM**) refers to how the web browser sees a web page -- as a ***hierarchy*** of objects, just like how files and folders are organized on your computer.

We use the term **DOM element** to refer to parts of the web page, like a paragraph (<p>), an image (<img>), a heading (<h1>), etc.



Each **element** can be a represented as a **JavaScript object** that contains LOTS of information about that element – for example, the text that appears inside the element, the background color of the element, etc.

Anatomy of an HTML Element

Opening tag: Closing tag:

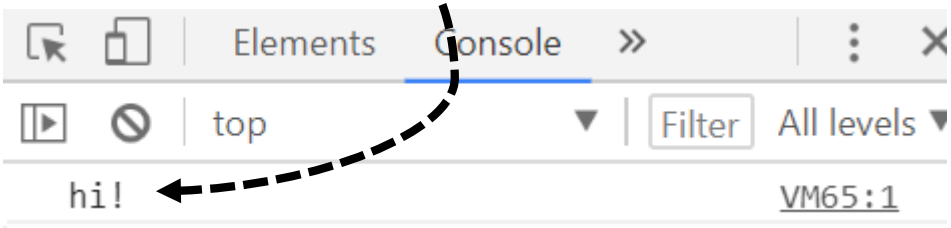
`<p id="main">` Hi, I'm text in a paragraph! `</p>`

- The p tag stands for ***paragraph***
- The id attribute gives the element a unique name – sort of like a social security number!
- So this paragraph has an ID of "main" and it's the *only* element on the page with that unique ID.

Displaying Data in the Console

The **console.log() function** can display values inside the web browser's JavaScript console, so you can see what's happening inside your code!

```
console.log("hi");
```




```
let greeting = "hi";
console.log(greeting);
```

You can **console.log()** lots of things -- raw values (like strings or numbers) or variable names!

Accessing Elements by their unique ID

```
<p id="main"> Hi, I'm text in a paragraph! </p>
```

```
document.getElementById("main")
```

Event Listener Example Code

```
                .addEventListener("click", funcName);

function funcName () {
    console.log( "You clicked!" );
}
```

Displaying Data inside Elements on the Web Page

```
                .textContent = "New text!";
```

You can access the **textContent** property of *any* **DOM element**.

For example:

```
document.body.textContent = "New text!";

document.getElementById("main").textContent = "New!";
```