HTTP

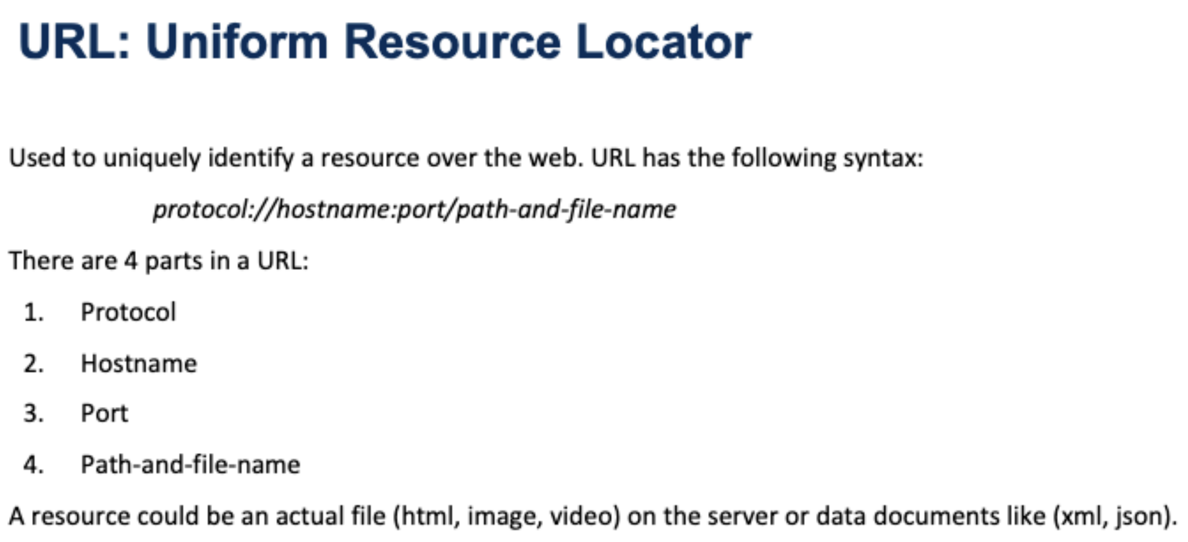
Response Status Code

The first line of the response message contains the response status code.
The status code is a 3-digit number:
1xx (Informational): Request received, server is continuing the process.
2xx (Success): The request was successfully received, understood, accepted and serviced. **3xx (Redirection):** Further action must be taken in order to complete the request.

4xx (Client Error): The request contains bad syntax or cannot be understood. **5xx (Server Error):** The server failed to fulfill an apparently valid request.

Common Status Codes

200 OK: The request is fulfilled.
301: Found & Redirect, but the new location is not temporary.
302 Found & Redirect (or Move Temporarily): Same as 301, but the new location is temporarily in nature. Client should update the references.
400 Bad Request: Server could not interpret or understand the request, probably syntax error in the request message. **401** Authentication Required: The requested resource is protected, and require client's credential (username/password). **403** Forbidden/Unauthorized: Server refuses to supply the resource, regardless of identity of client. In this case, client is not authorized to access the resource.
404 Not Found: The requested resource cannot be found in the server.
500 Internal Server Error: Caused by an error in the server-side program responding to the request.
501 Method Not Implemented: The request method used is invalid (method name is case sensitive so even misspelling could cause this.).



URL: Uniform Resource Locator

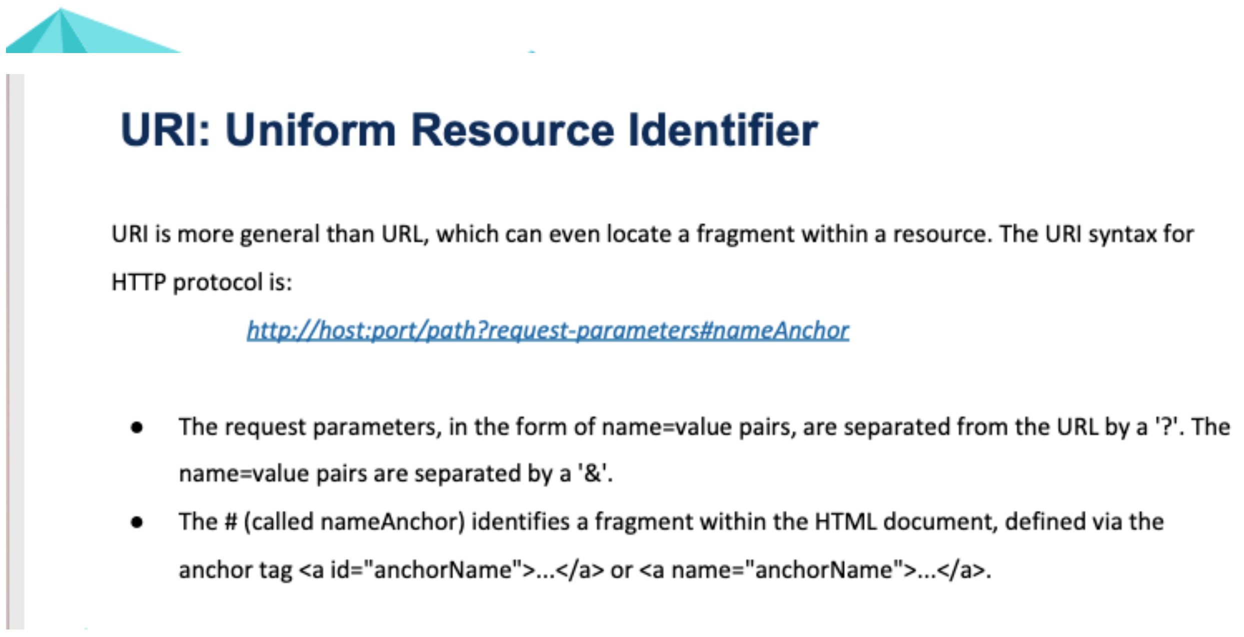
Used to uniquely identify a resource over the web. URL has the following syntax:

protocol://hostname:port/path-and-file-name

There are 4 parts in a URL:

1. Protocol
2. Hostname
3. Port
4. Path-and-file-name

A resource could be an actual file (html, image, video) on the server or data documents like (xml, json).



URI: Uniform Resource Identifier

URI is more general than URL, which can even locate a fragment within a resource. The URI syntax for HTTP protocol is:

http://host:port/path?request-parameters#nameAnchor

- The request parameters, in the form of name=value pairs, are separated from the URL by a '?'. The name=value pairs are separated by a '&'.
- The # (called nameAnchor) identifies a fragment within the HTML document, defined via the anchor tag <a id="anchorName">...</a> or <a name="anchorName">...</a>.

GET: Used for retrieving a web resource.

POST: Used for creating/posting new data up to the web server.

PUT: Used for updating a resource on the server.

DELETE: Used for deleting a resource on the server.

OPTIONS: Used for getting all methods accepted by server.

response status code

3 digit number

request header

response header

Git

git clone: to clone a repository

git add: to stage changes

git add --all: to stage new files

git commit -m "message" to commit changes

git commit -am: to stage changes and perform commit

git checkout： 切换分支

git checkout -b创建新分支并切换

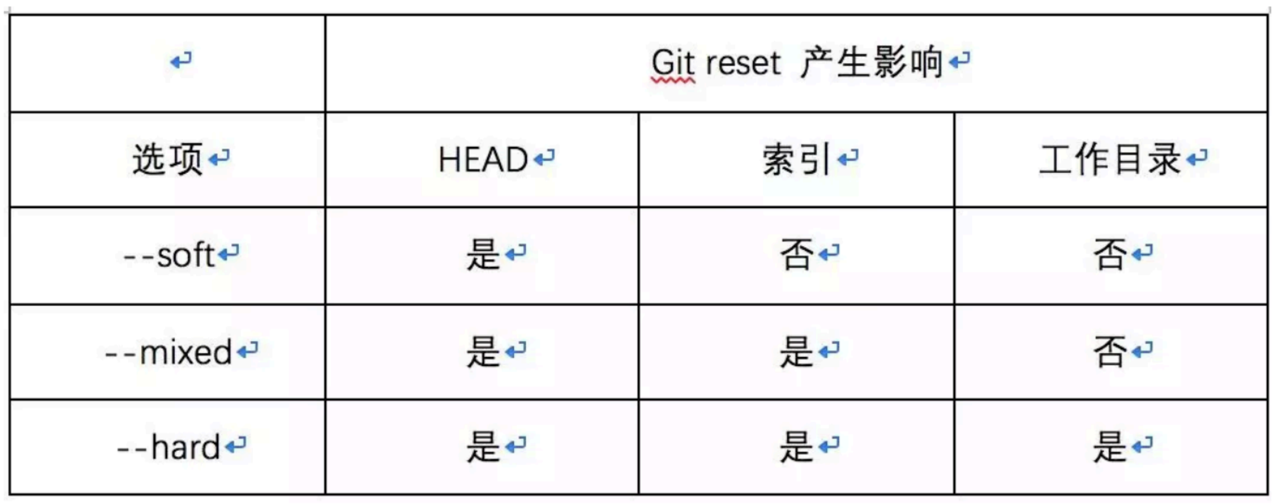
git pull

git merge

git fetch <remote_repo_name> <remote_branch_name>:<local_branch_name>——可以认为git pull是git fetch和git merge两个步骤的结合。

git reset --hard: 移动head指向的分支， 移除当前分支的一些提交。 HEAD引用指向给定提交，索引（暂存区）内容和工作目录内容都会变给定提交时的状态。也就是在给定提交后所修改的内容都会丢失(新文件会被删除，不在工作目录中的文件恢复，未清除回收站的前提)。

用表格看起来会更清楚些：



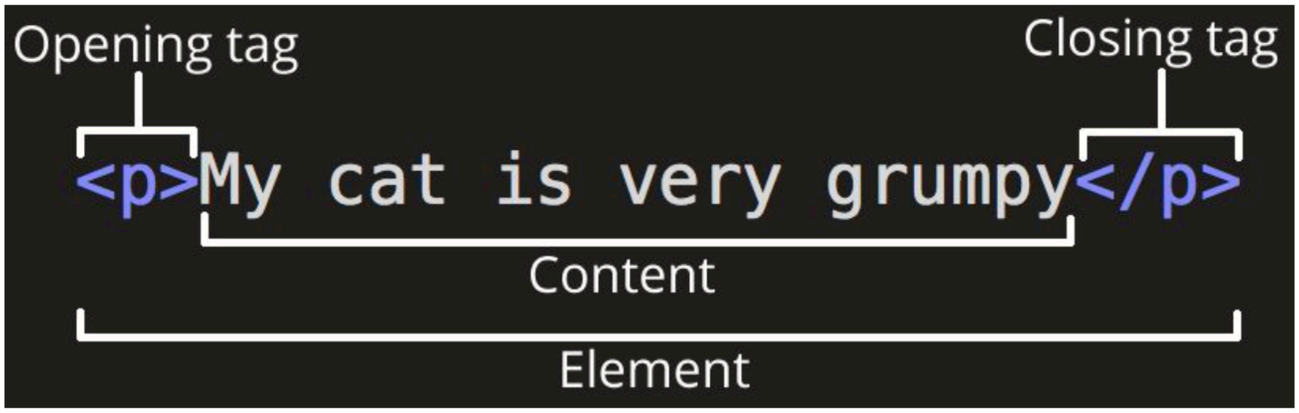
	Git reset 产生影响		
选项	HEAD	索引	工作目录
--soft	是	否	否
--mixed	是	是	否
--hard	是	是	是

HTML

hypertex markuo language

not programming language, it is markup language to tell browaer how to structure the webpage

HTML consists of a series of elements, which you use to enclose, wrap, or mark up different parts of the content to make it appear or act a certain way



elements

- nested
- block elements: a new line.
  - Paragraphs
  - lists
  - navigation menus
  - footers
- inline elements: contained within bolck-level elements, surround only small parts of the documnts content. Not entire paragraphs and grouping of content.

  no new line.

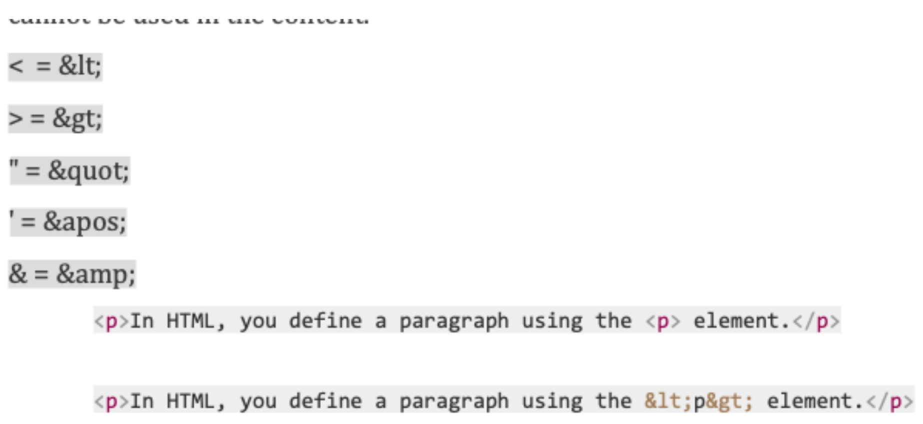
- ***empty elements: Some elements consist only of a single tag, which is usually used to insert/embed something in the document at the place it is included***

attribues

attributes contain extra information about the element which you don't want to appear in the actual content

boolean attributes: some attributes without values. they only have one value, which is same as the attributes name.

either single or double quotes, but alway double. can omit quotes, but when multiple attributes will break



cannot be used in the content.

< = &lt;

> = &gt;

" = &quot;

' = &apos;

& = &amp;

```
<p>In HTML, you define a paragraph using the <p> element.</p>

<p>In HTML, you define a paragraph using the &lt;p&gt; element.</p>
```

meta

provides metadata about the page

table

Tr: a row in a table

td：a cell in a table

th：describe the cell is at the top of a column

Colgroup: a group of colgroup

tfoot

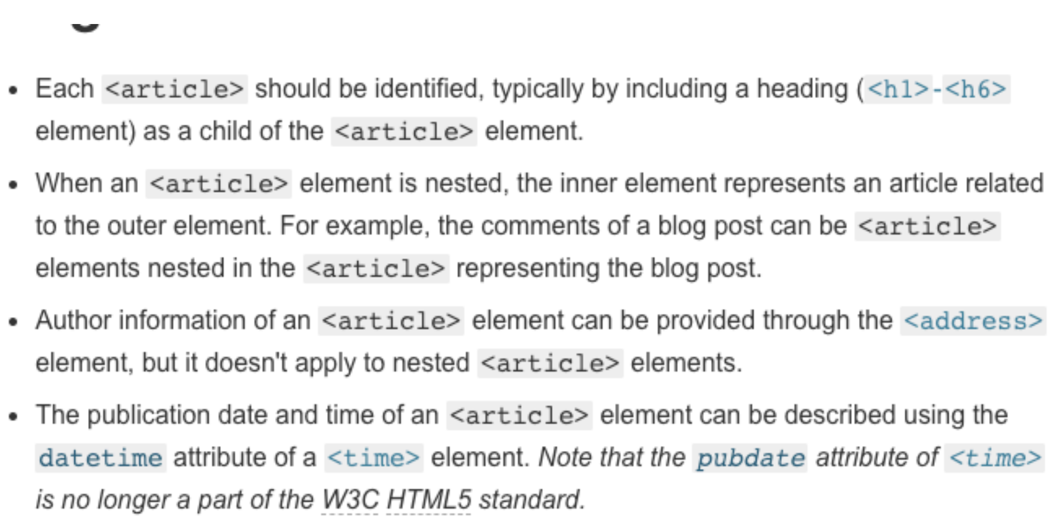
form

action

method

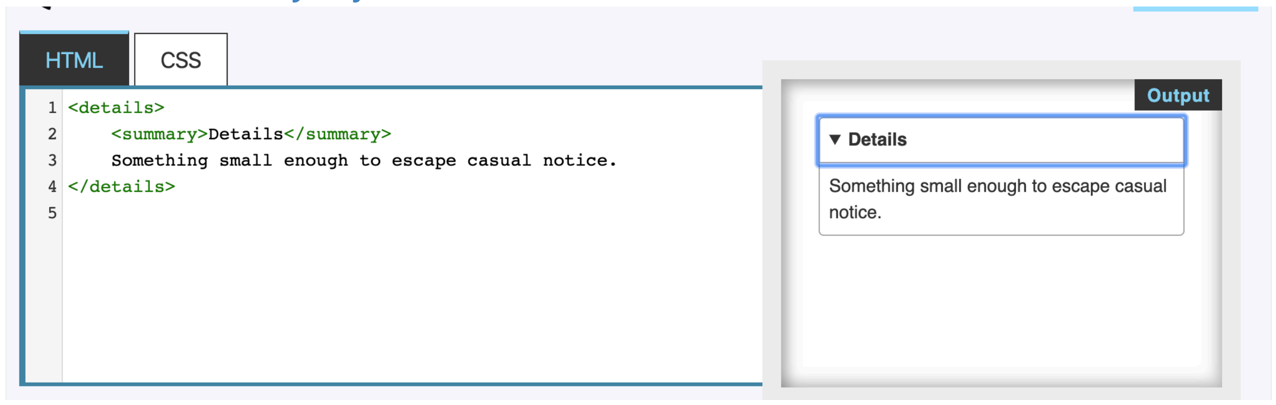
datalist包含了一组option

article intended to be independently distributable or reusable (e.g., in syndication). Examples include: a forum post, a magazine or newspaper article, or a blog entry.



- Each `<article>` should be identified, typically by including a heading (`<h1>`-`<h6>` element) as a child of the `<article>` element.
- When an `<article>` element is nested, the inner element represents an article related to the outer element. For example, the comments of a blog post can be `<article>` elements nested in the `<article>` representing the blog post.
- Author information of an `<article>` element can be provided through the `<address>` element, but it doesn't apply to nested `<article>` elements.
- The publication date and time of an `<article>` element can be described using the `datetime` attribute of a `<time>` element. *Note that the `pubdate` attribute of `<time>` is no longer a part of the W3C HTML5 standard.*

detail contains summary: default is close



```html
<details>
    <summary>Details</summary>
    Something small enough to escape casual notice.
</details>
```

```
▼ Details
Something small enough to escape casual notice.
```

header contains some heading elements but also a logo, a search form, an author name, and other elements

footer represents a footer for its nearest sectioning content or sectioning root element. A footer typically contains information about the author of the section, copyright data or links to related documents.

nav represents a section of a page whose purpose is to provide navigation links, either within the current document or to other documents. Common examples of navigation sections are menus, tables of contents, and indexes.

ol order html列表

li 列表里面的每一行, item in a list

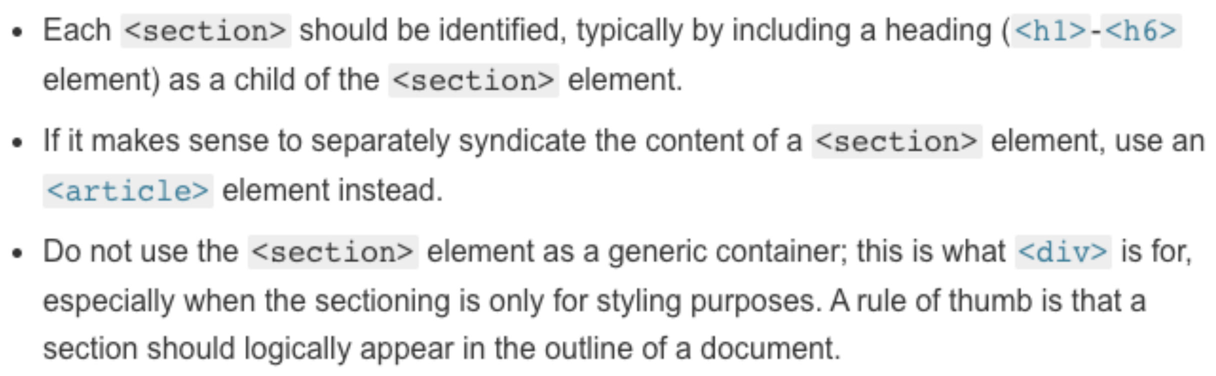
ul unorder html list

section Each

should be identified, typically by including a heading (

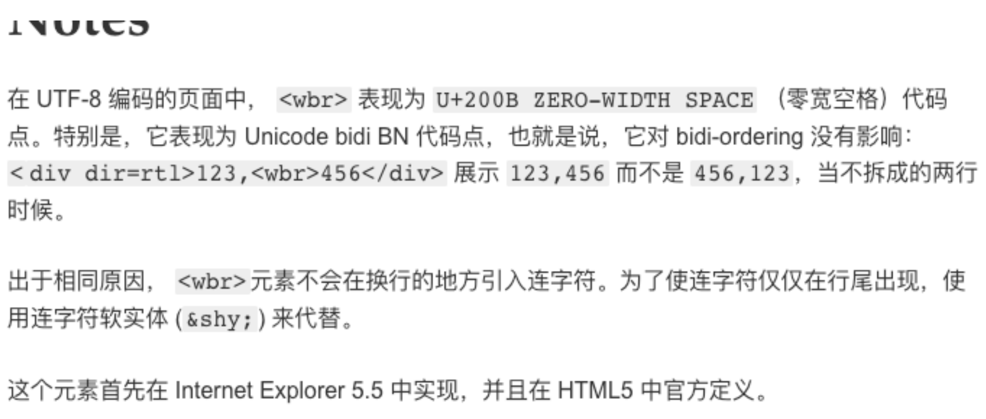
–

element) as a child of the
element.



- Each `<section>` should be identified, typically by including a heading (`<h1>`-`<h6>` element) as a child of the `<section>` element.
- If it makes sense to separately syndicate the content of a `<section>` element, use an `<article>` element instead.
- Do not use the `<section>` element as a generic container; this is what `<div>` is for, especially when the sectioning is only for styling purposes. A rule of thumb is that a section should logically appear in the outline of a document.

wbr represents a word break opportunity—a position within text where the browser may optionally break a line, though its line-breaking rules would not otherwise create a break at that location.

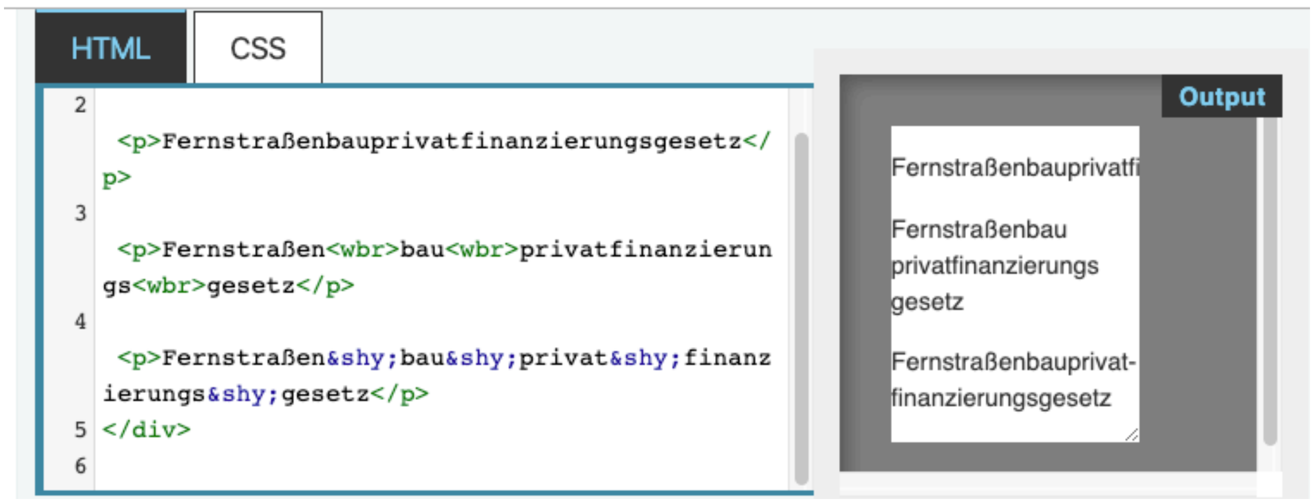


Notes

在 UTF-8 编码的页面中， `<wbr>` 表现为 `U+200B ZERO-WIDTH SPACE` （零宽空格）代码点。特别是，它表现为 Unicode bidi BN 代码点，也就是说，它对 bidi-ordering 没有影响：`<div dir=rtl>123,<wbr>456</div>` 展示 `123,456` 而不是 `456,123`，当不拆成的两行时候。

出于相同原因， `<wbr>`元素不会在换行的地方引入连字符。为了使连字符仅仅在行尾出现，使用连字符软实体 (`&shy;`) 来代替。

这个元素首先在 Internet Explorer 5.5 中实现，并且在 HTML5 中官方定义。



HTML CSS

```
  <p>Fernstraßenbauprivatfinanzierungsgesetz</p>
  <p>Fernstraßen<wbr>bau<wbr>privatfinanzierungs<wbr>gesetz</p>
  <p>Fernstraßen&shy;bau&shy;privat&shy;finanzierungs&shy;gesetz</p>
</div>
```

Output

```
Fernstraßenbauprivatfi

Fernstraßenbau
privatfinanzierungs
gesetz

Fernstraßenbauprivat-
finanzierungsgesetz
```

audio is used to embed sound content in documents. It may contain one or more audio sources, represented using the src attribute or the element: the browser will choose the most suitable one. It can also be the destination for streamed media, using a MediaStream.

video embeds a media player which supports video playback into the document. You can use *for audio content as well, but the element may provide a more appropriate user experience.*

svg

Scalable Vector Graphics (SVG) is an XML-based markup language for describing two dimensional based vector graphics. SVG is essentially to graphics what HTML is to text

canvas

The canvas element is part of HTML5 and allows for dynamic, scriptable rendering of 2D and 3D shapes and bitmap images. It is a low level, procedural model that updates a bitmap and does not have a built-in scene graph. It provides an empty graphic zone on which specific JavaScript APIs can draw (such as Canvas 2D or WebGL).

Web Components

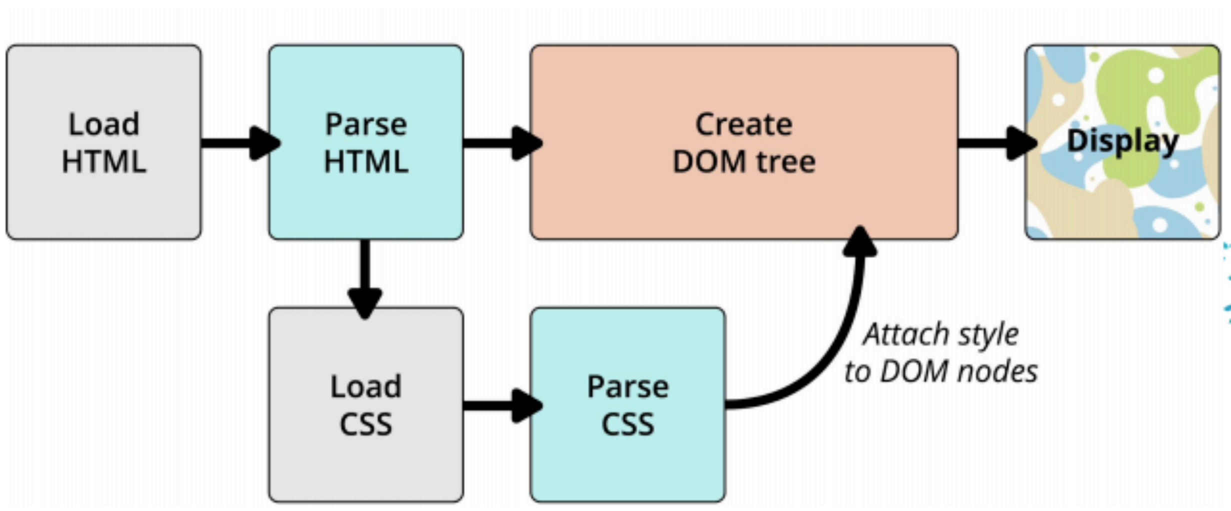
Web Components is a suite of different technologies allowing you to create reusable custom user interface components — with their functionality encapsulated away from the rest of your code — and utilize them in your web apps.

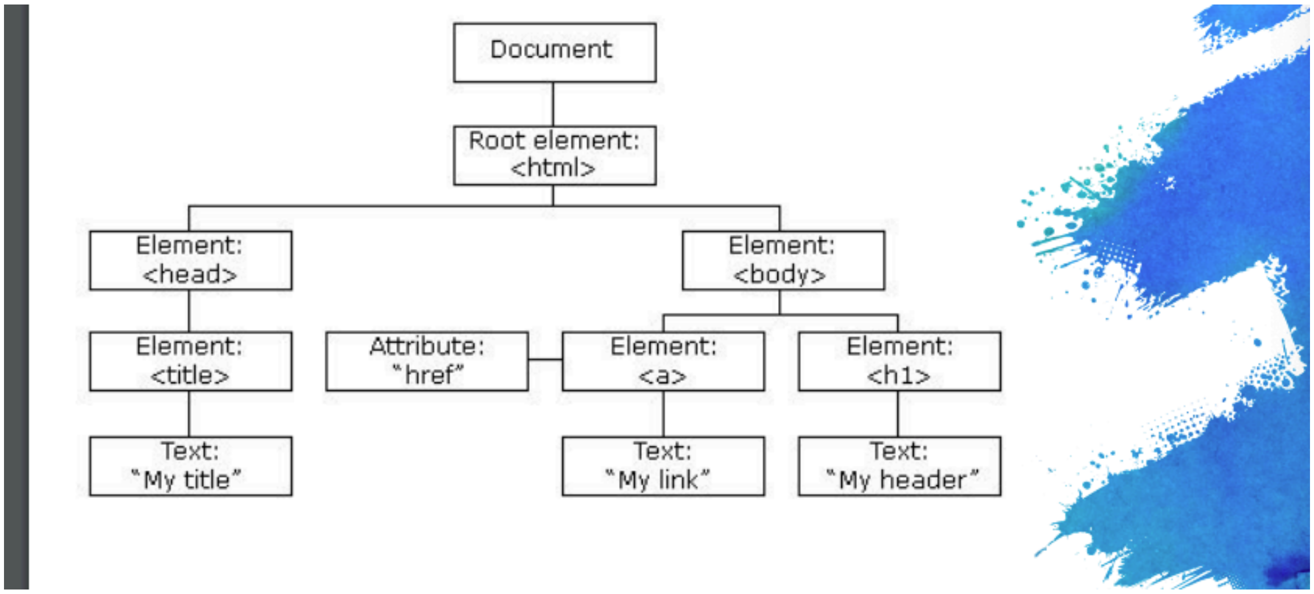
CSS

presentation of a document written in HTML XML

how elements be rendered on screen, paper, speach or other media

DOM document object model





each in property in the set along with its value is calles a declaration

a block of declarations is called declaration block

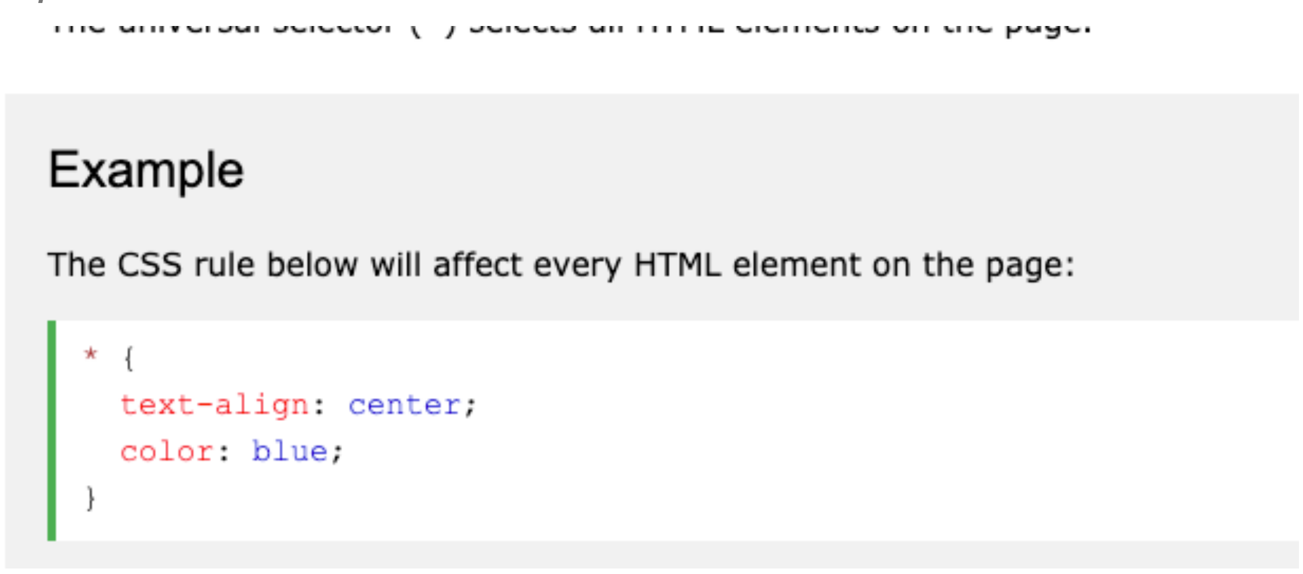
decalaration block along with selector is a css rulers or css rule

external stylesheet

internal stylesheet

inline styles

simple selector


The universal selector (*) selects all HTML elements on the page.

Example

The CSS rule below will affect every HTML element on the page:

```
* {
  text-align: center;
  color: blue;
}
```

descendant selector

combines two selectors such that elements matched by the second selector are selected if they have an ancestor (parent, parent's parent, parent's parent's parent, etc) element matching the first selector. Selectors that utilize a descendant combinator are called descendant selectors.

child selector >

It matches only those elements matched by the second selector that are the direct children of elements matched by the first.

adjacent sibling selector +

separates two selectors and matches the second element only if it immediately follows the first element, and both are children of the same parent element.

一定要直接相邻，中间不插别的元素

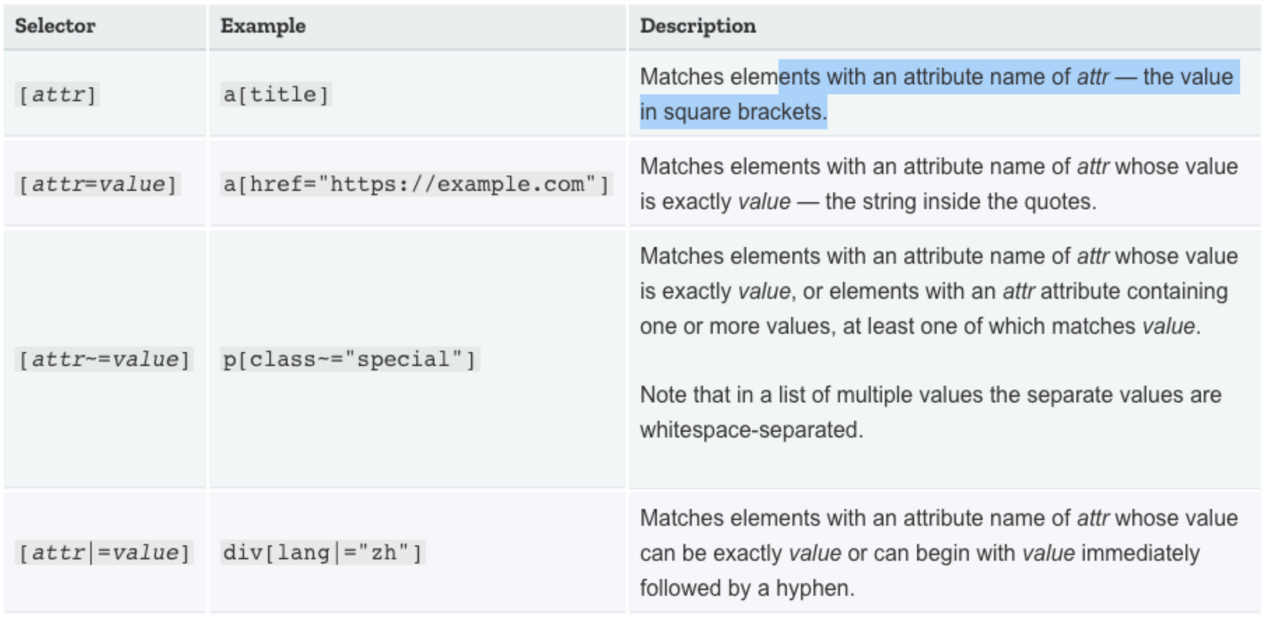
general sibling selector ~

separates two selectors and matches the second element only if it follows the first element (though not necessarily immediately), and both are children of the same parent element.

attribute selectors

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Attribute_selectors

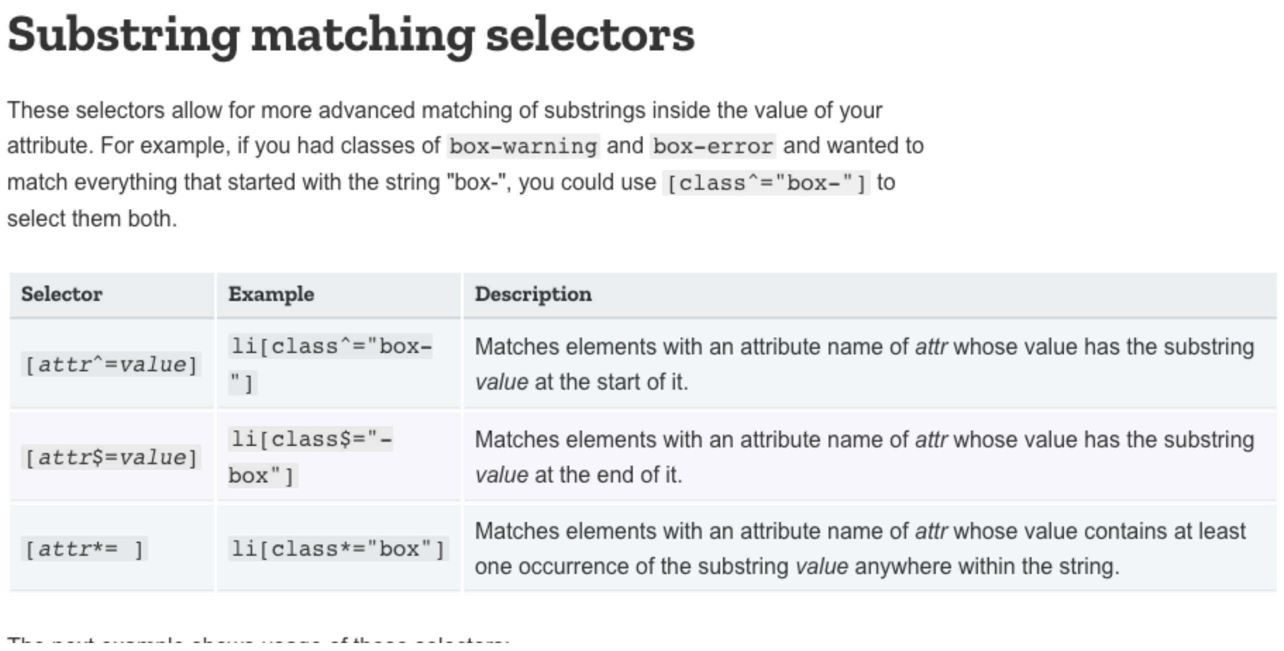
In CSS you can use attribute selectors to target elements with certain attributes



Selector	Example	Description
`[attr]`	`a[title]`	Matches elements with an attribute name of *attr* — the value in square brackets.
`[attr=value]`	`a[href="https://example.com"]`	Matches elements with an attribute name of *attr* whose value is exactly *value* — the string inside the quotes.
`[attr~=value]`	`p[class~="special"]`	Matches elements with an attribute name of *attr* whose value is exactly *value*, or elements with an *attr* attribute containing one or more values, at least one of which matches *value*. Note that in a list of multiple values the separate values are whitespace-separated.
`[attr|=value]`	`div[lang|="zh"]`	Matches elements with an attribute name of *attr* whose value can be exactly *value* or can begin with *value* immediately followed by a hyphen.

attr~= 含有其中的一个完全相同的value

attr｜=其值可以完全是value，也可以以value开头，后跟连字符。



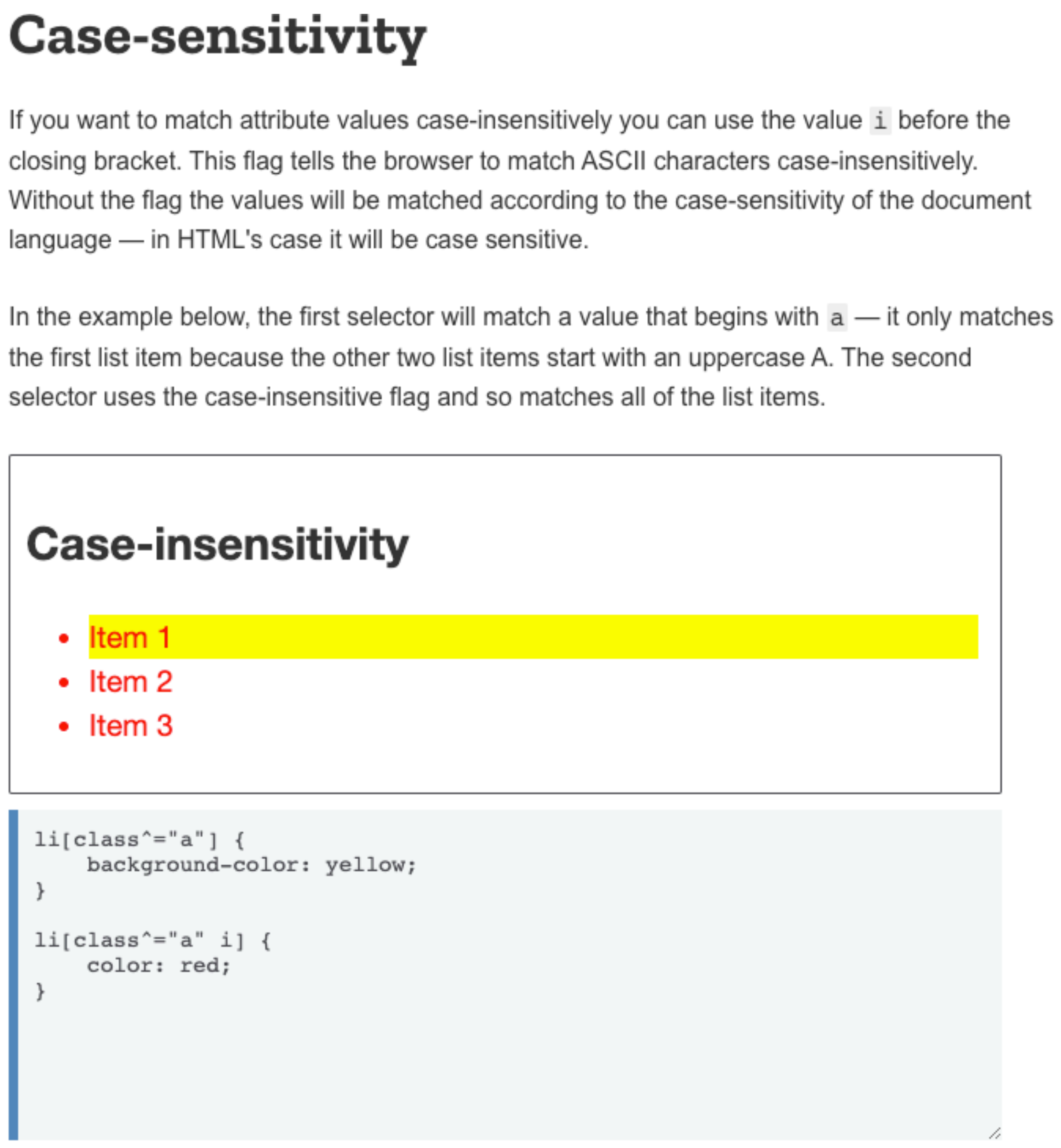
Substring matching selectors

These selectors allow for more advanced matching of substrings inside the value of your attribute. For example, if you had classes of `box-warning` and `box-error` and wanted to match everything that started with the string "box-", you could use `[class^="box-"]` to select them both.

Selector	Example	Description
`[attr^=value]`	`li[class^="box-"]`	Matches elements with an attribute name of *attr* whose value has the substring *value* at the start of it.
`[attr$=value]`	`li[class$="-box"]`	Matches elements with an attribute name of *attr* whose value has the substring *value* at the end of it.
`[attr*= ]`	`li[class*="box"]`	Matches elements with an attribute name of *attr* whose value contains at least one occurrence of the substring *value* anywhere within the string.

The next example shows usage of these selectors:

不区分大小写在值后面添加i



Case-sensitivity

If you want to match attribute values case-insensitively you can use the value `i` before the closing bracket. This flag tells the browser to match ASCII characters case-insensitively. Without the flag the values will be matched according to the case-sensitivity of the document language — in HTML's case it will be case sensitive.

In the example below, the first selector will match a value that begins with `a` — it only matches the first list item because the other two list items start with an uppercase A. The second selector uses the case-insensitive flag and so matches all of the list items.

Case-insensitivity

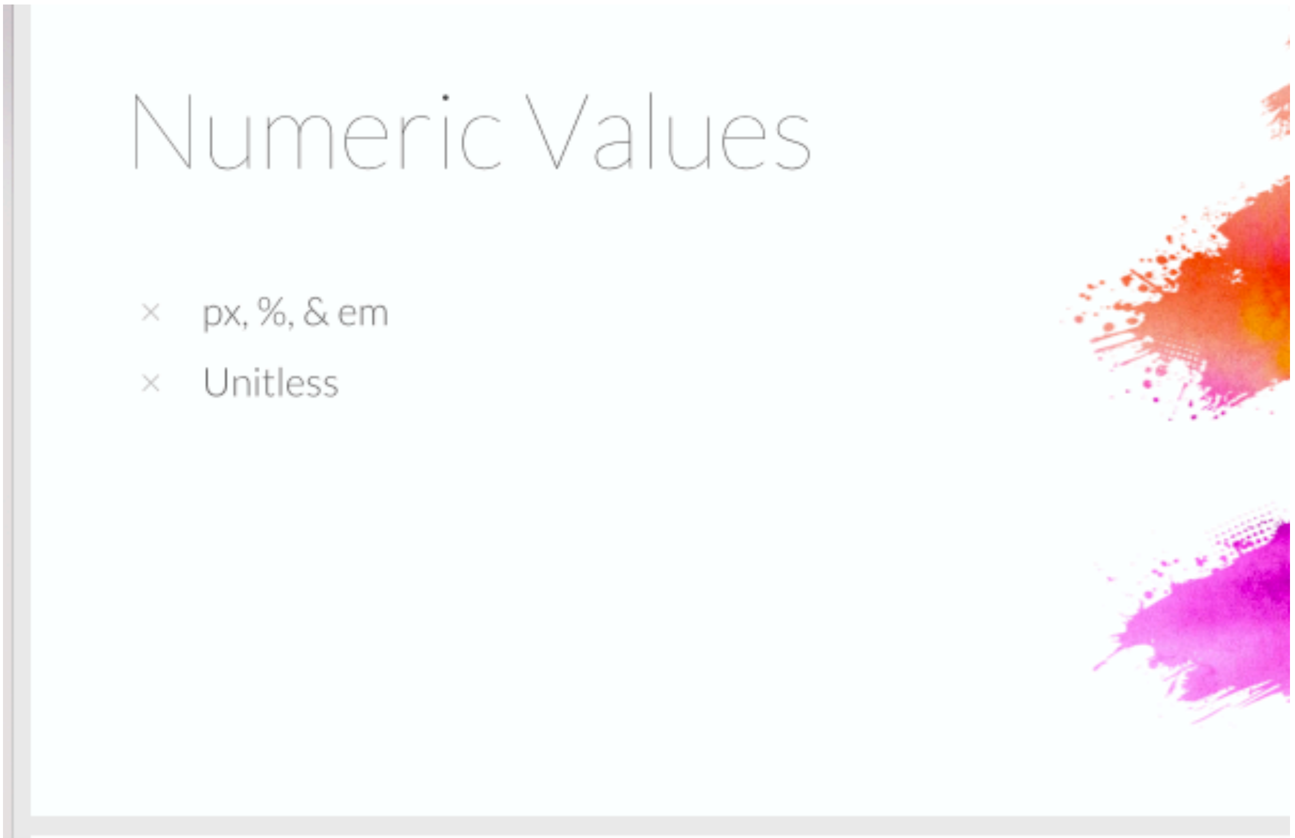
- Item 1
- Item 2
- Item 3

```
li[class^="a"] {
    background-color: yellow;
}

li[class^="a" i] {
    color: red;
}
```

mutiple selectors

逗号分割

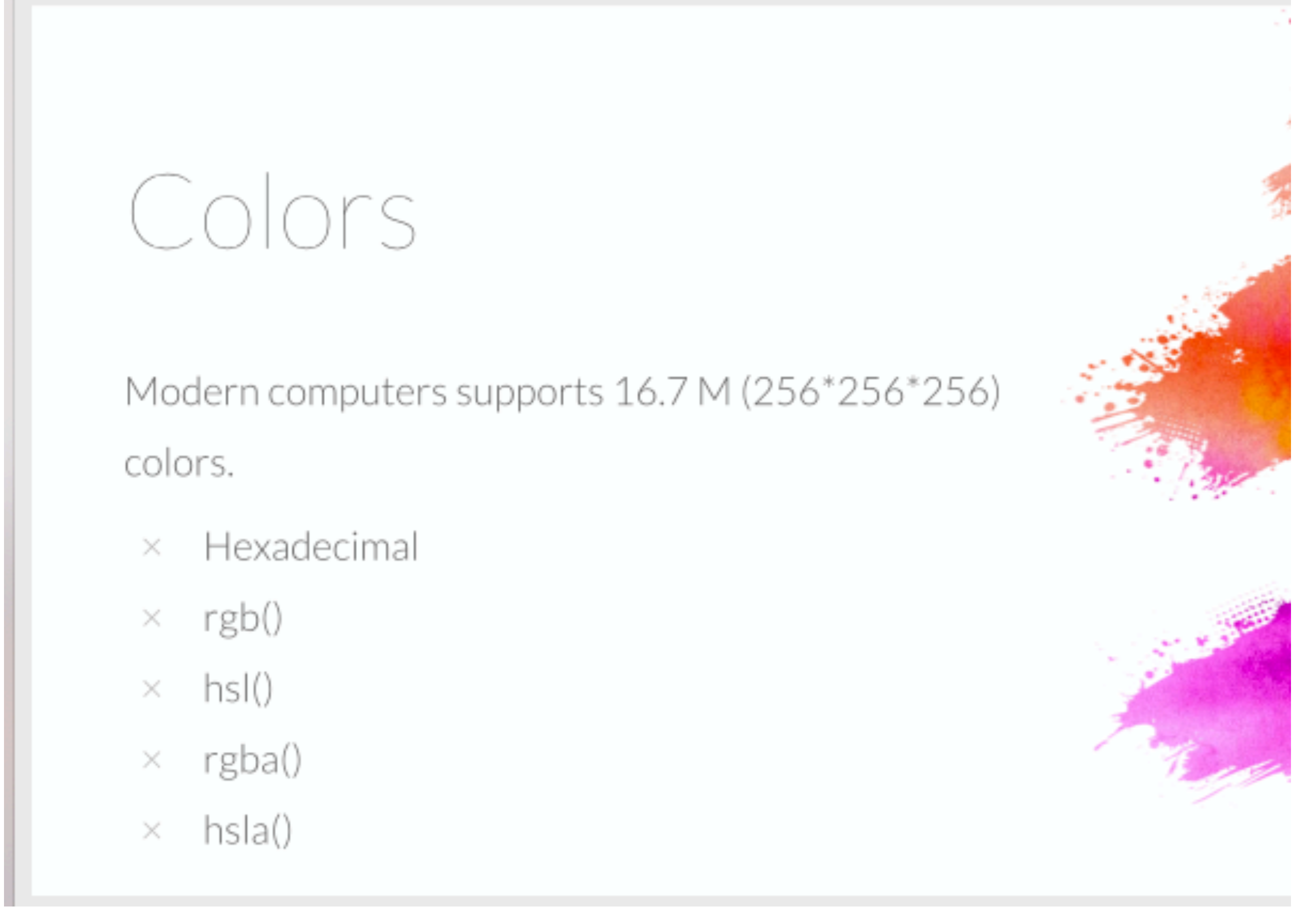
values



Numeric Values

× px, %, & em

× Unitless



Colors

Modern computers supports 16.7 M (256*256*256) colors.

× Hexadecimal

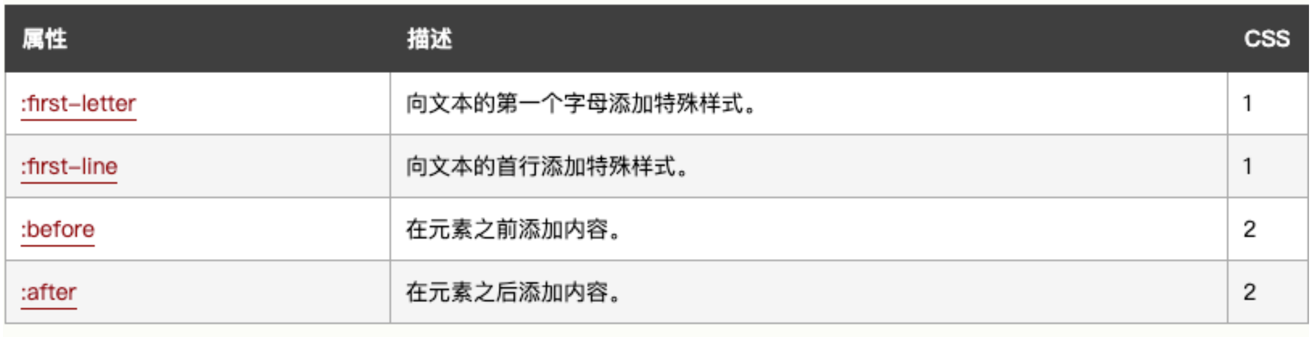
× rgb()

× hsl()

× rgba()

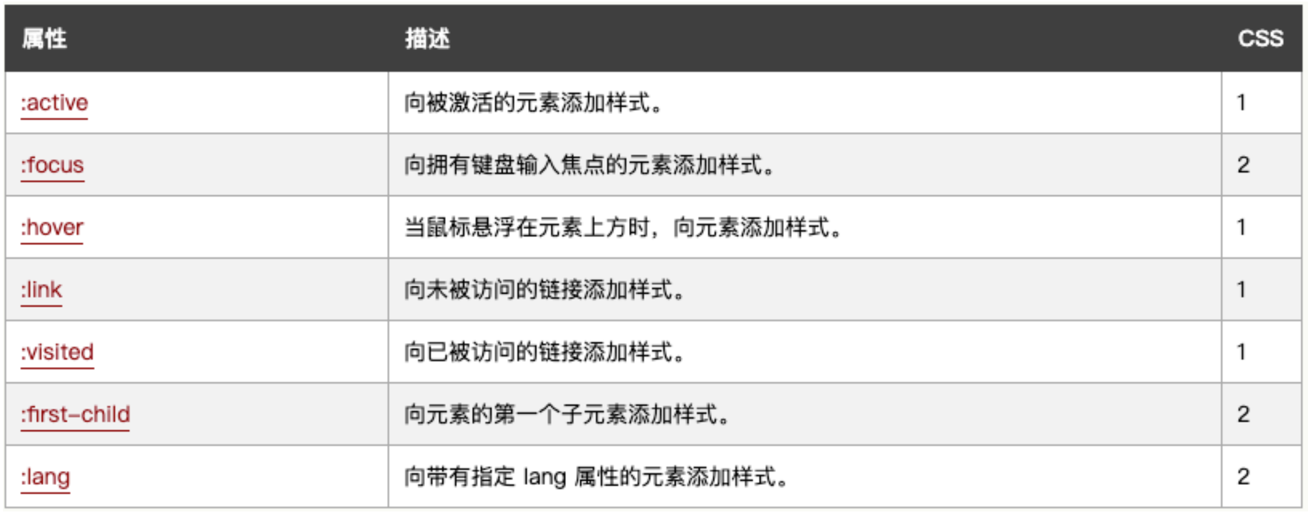
× hsla()

css伪元素



属性	描述	CSS
:first-letter	向文本的第一个字母添加特殊样式。	1
:first-line	向文本的首行添加特殊样式。	1
:before	在元素之前添加内容。	2
:after	在元素之后添加内容。	2

css伪类



属性	描述	CSS
:active	向被激活的元素添加样式。	1
:focus	向拥有键盘输入焦点的元素添加样式。	2
:hover	当鼠标悬浮在元素上方时，向元素添加样式。	1
:link	向未被访问的链接添加样式。	1
:visited	向已被访问的链接添加样式。	1
:first-child	向元素的第一个子元素添加样式。	2
:lang	向带有指定 lang 属性的元素添加样式。	2

type of box

display**

block: The element generates a block element box, generating line breaks both before and after the element when in the normal flow.

前后都会有换行符

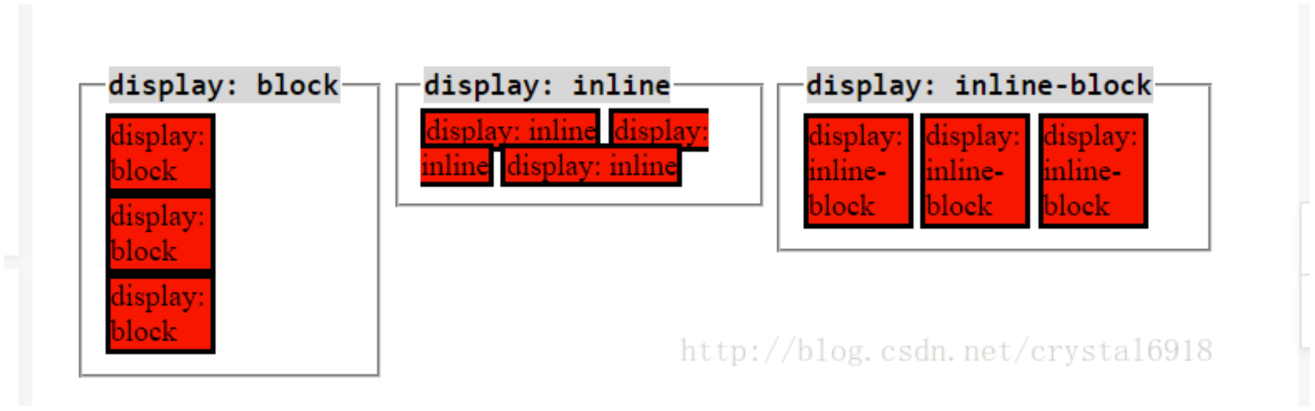
inline： The element generates one or more inline element boxes that do not generate line breaks before or after themselves. In normal flow, the next element will be on the same line if there is space

Inline-block

Compared to `display: inline`, the major difference is that `display: inline-block` allows to set a width and height on the element.

Also, with `display: inline-block`, the top and bottom margins/paddings are respected, but with `display: inline` they are not.

Compared to `display: block`, the major difference is that `display: inline-block` does not add a line-break after the element, so the element can sit next to other elements.



overflow

property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

- `visible`*-默认。溢出没有被修剪。内容在元素框外渲染*
- `hidden` *-溢出被裁剪，其余内容将不可见*
- `scroll` *-剪辑了溢出，并添加了滚动条以查看其余内容*
- `auto`*-与相似*`scroll`*，但仅在必要时添加滚动条*

inheritance

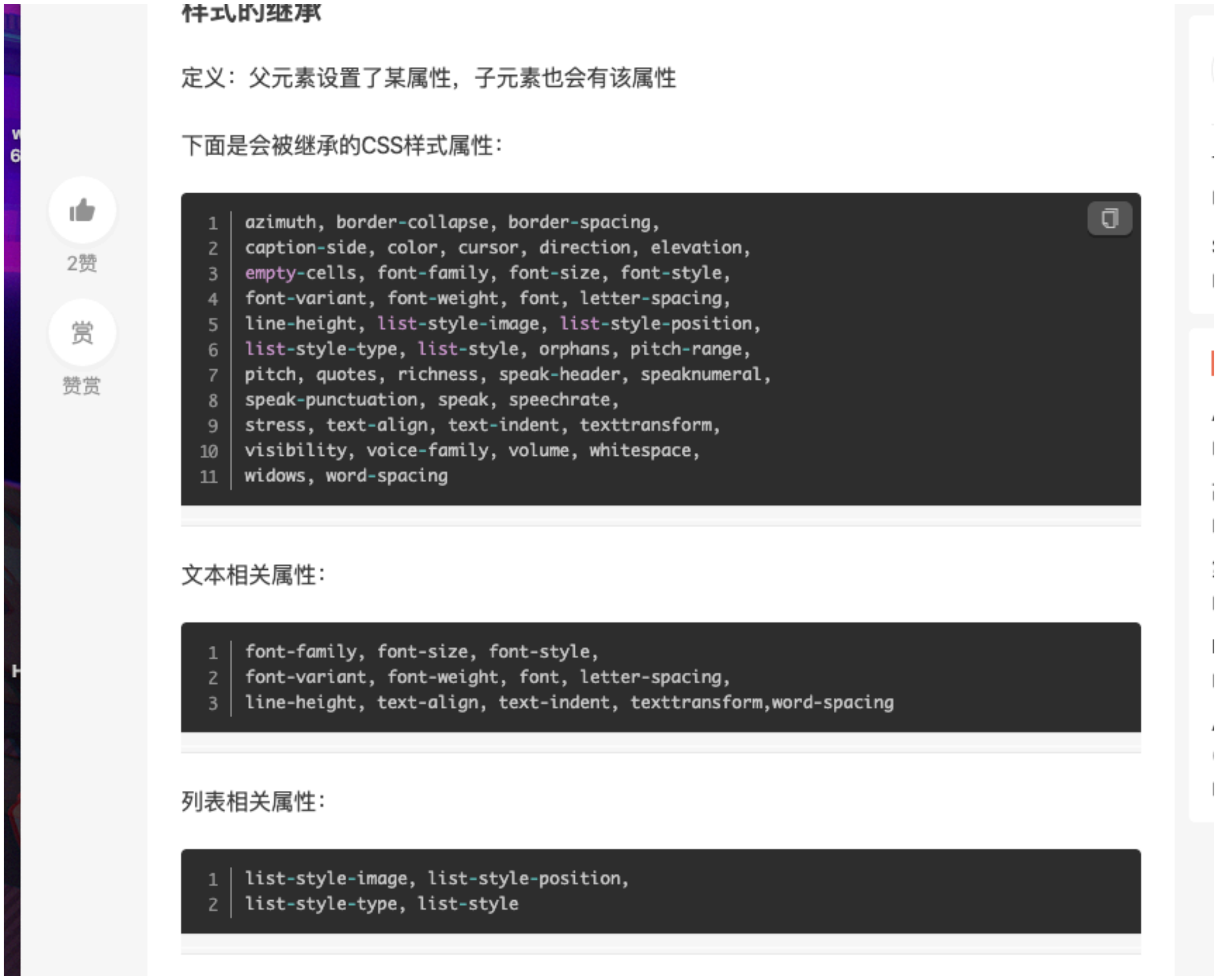
A typical example of an inherited property is the `color` property. Given the style rules:

```
p { color: green; }
```

... and the markup:

```
<p>This paragraph has <em>emphasized text</em> in it.</p>
```

... the words "emphasized text" will appear green, since the `em` element has inherited the value of the `color` property from the `p` element. It does *not* get the initial value of the property (which is the color that is used for the root element when the page specifies no color).



样式的继承

定义：父元素设置了某属性，子元素也会有该属性

下面是会被继承的CSS样式属性：

```
azimuth, border-collapse, border-spacing,
caption-side, color, cursor, direction, elevation,
empty-cells, font-family, font-size, font-style,
font-variant, font-weight, font, letter-spacing,
line-height, list-style-image, list-style-position,
list-style-type, list-style, orphans, pitch-range,
pitch, quotes, richness, speak-header, speaknumeral,
speak-punctuation, speak, speechrate,
stress, text-align, text-indent, texttransform,
visibility, voice-family, volume, whitespace,
widows, word-spacing
```

文本相关属性：

```
font-family, font-size, font-style,
font-variant, font-weight, font, letter-spacing,
line-height, text-align, text-indent, texttransform,word-spacing
```

列表相关属性：

```
list-style-image, list-style-position,
list-style-type, list-style
```

The `initial` CSS keyword applies the [initial (or default) value]() of a property to an element. This initial value is set by the browser. It can be applied to any CSS property. This includes the CSS shorthand [`all`](), with which `initial` can be used to restore all CSS properties to their initial state.

！import的优先级最高

position： https://developer.mozilla.org/en-US/docs/Web/CSS/position

static：

该关键字指定元素使用正常的布局行为，即元素在文档常规流中当前的布局位置。此时 `top`, `right`, `bottom`, `left` 和 `z-index` 属性无效。

absolute：对父元素

元素会被移出正常文档流，并不为元素预留空间，通过指定元素相对于最近的非 static 定位祖先元素的偏移，来确定元素位置。绝对定位的元素可以设置外边距（margins），且不会与其他边距合并

relative：

该关键字下，元素先放置在未添加定位时的位置，再在不改变页面布局的前提下调整元素位置（因此会在此元素未添加定位时所在位置留下空白）。position:relative 对 table--group, table-row, table-column, table-cell, table-caption 元素无效。*

sticky：

元素根据正常文档流进行定位，然后相对它的最近滚动祖先（nearest scrolling ancestor）和 containing block (最近块级祖先 nearest block-level ancestor)，包括table-related元素，基于`top`, `right`, `bottom`, 和 `left` 的值进行偏移。偏移值不会影响任何其他元素的位置。

该值总是创建一个新的层叠上下文（stacking context）。注意，一个sticky元素会“固定”在离它最近的一个拥有“滚动机制”的祖先上（当该祖先的`overflow` 是 `hidden`, `scroll`, `auto`, 或 `overlay`时），即便这个祖先不是真的滚动祖先。这个阻止了所有“sticky”行为（详情见Github issue on W3C CSSWG）。

fixed：　对窗口

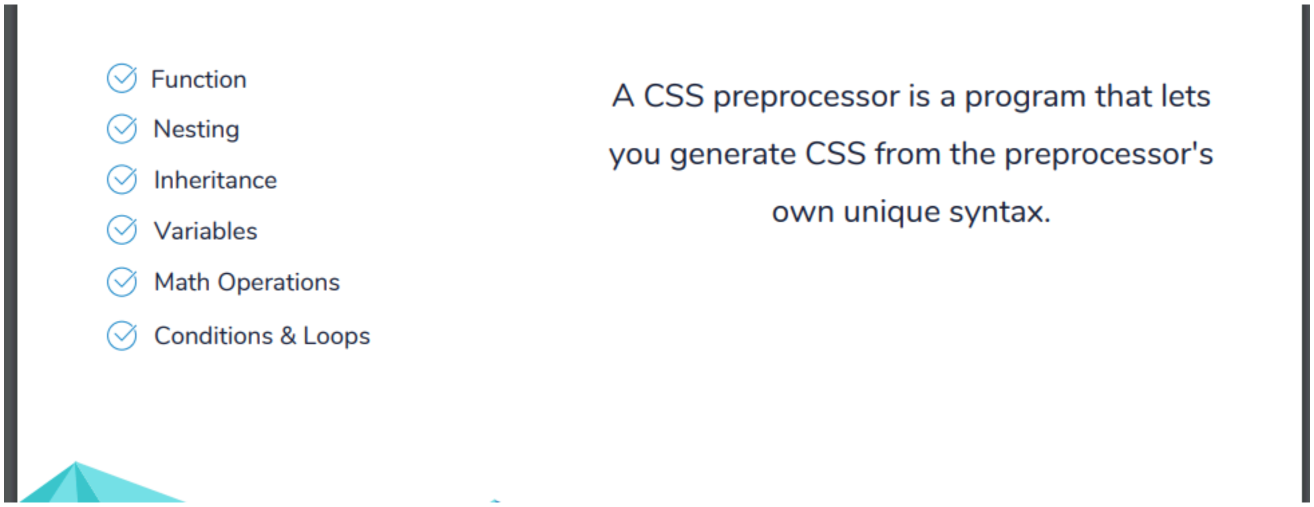
元素会被移出正常文档流，并不为元素预留空间，而是通过指定元素相对于屏幕视口（viewport）的位置来指定元素位置。元素的位置在屏幕滚动时不会改变。打印时，元素会出现在的每页的固定位置。`fixed` 属性会创建新的层叠上下文。当元素祖先的`transform`, `perspective` 或 `filter` 属性非 `none` 时，容器由视口改为该祖先。

CSS animation

https://developer.mozilla.org/en-US/docs/Web/CSS/animation

SCSS

https://sass-lang.com/documentation/syntax



- Function
- Nesting
- Inheritance
- Variables
- Math Operations
- Conditions & Loops

A CSS preprocessor is a program that lets you generate CSS from the preprocessor's own unique syntax.

JavaScript

JavaScript is a loosely typed or a dynamic language. That means you don't have to declare the type of a variable ahead of time. The type will get determined automatically while the program is being processed. That also means that you can have the same variable as different types.

Boolean

null

undefined

number

string

object

array

date

attention

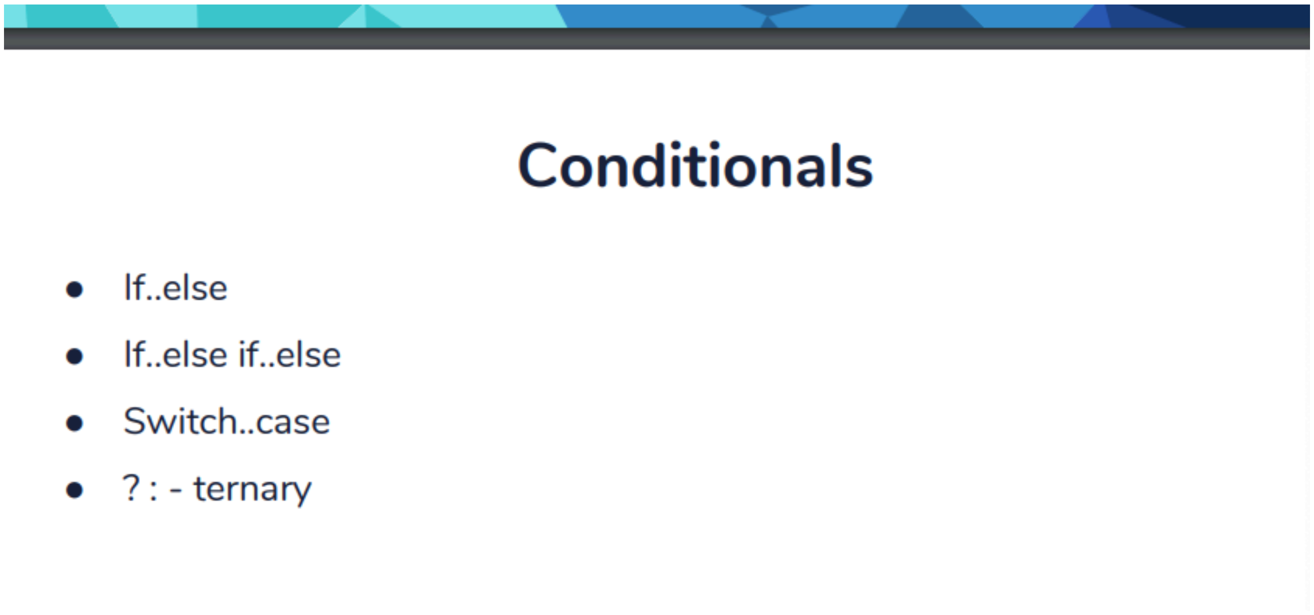
null is a object, just has no value

Undeified type just undefined

在if condition中表现几乎一样

===strict equality

!== strict-non-equality



Conditionals

- If..else
- If..else if..else
- Switch..case
- ? : - ternary

prototype

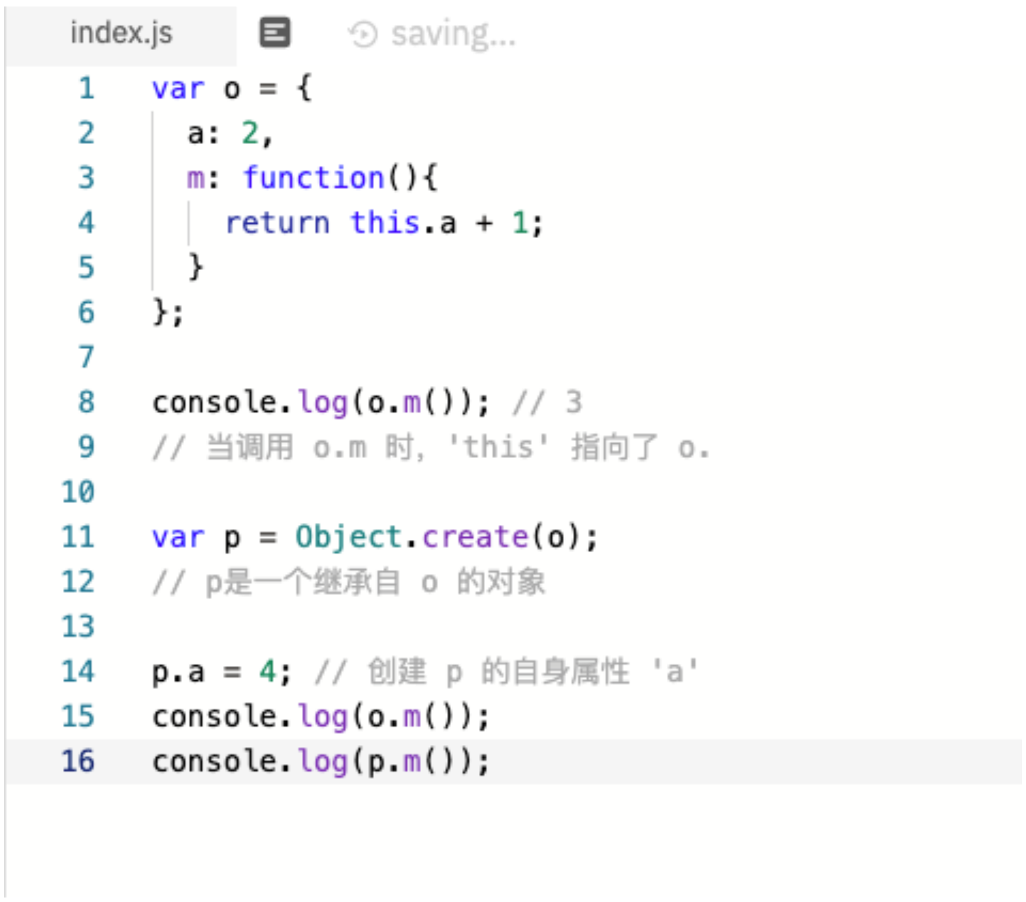
就是一个东西的原型

A prototype is a model that displays the appearance and behavior of an application or product early in the development lifecycle.

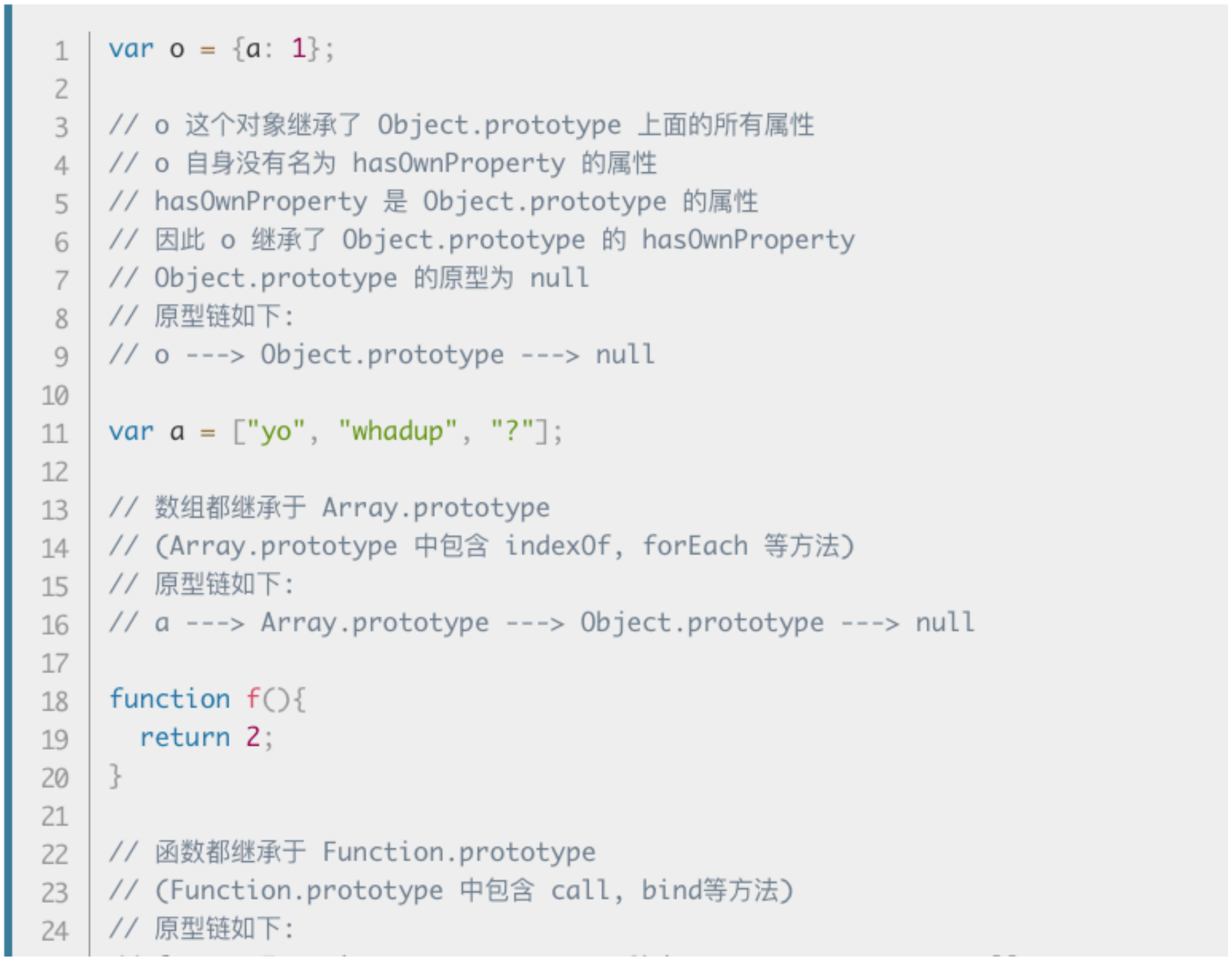
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Inheritance_and_the_prototype_chain

Here is what happens when trying to access a property.

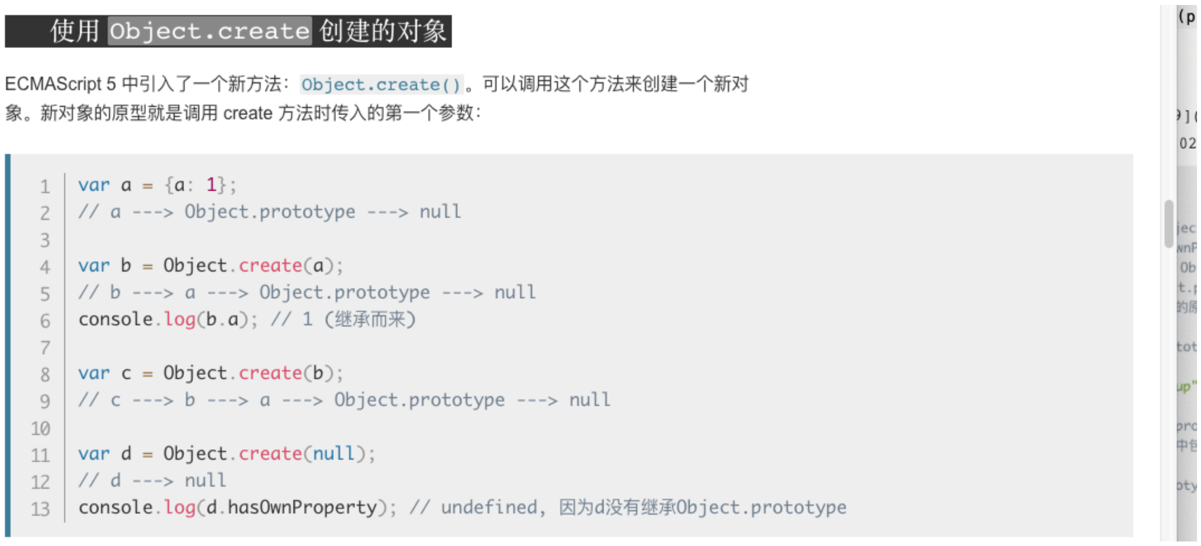
```javascript
// Let's create an object o from function f with its own properties a and b:
let f = function () {
   this.a = 1;
   this.b = 2;
}
let o = new f(); // {a: 1, b: 2}

// add properties in f function's prototype
f.prototype.b = 3;
f.prototype.c = 4;

// do not set the prototype f.prototype = {b:3,c:4}; this will break the prototype chain
// o.[[Prototype]] has properties b and c.
// o.[[Prototype]].[[Prototype]] is Object.prototype.
// Finally, o.[[Prototype]].[[Prototype]].[[Prototype]] is null.
// This is the end of the prototype chain, as null,
// by definition, has no [[Prototype]].
// Thus, the full prototype chain looks like:
// {a: 1, b: 2} ---> {b: 3, c: 4} ---> Object.prototype ---> null

console.log(o.a); // 1
// Is there an 'a' own property on o? Yes, and its value is 1.

console.log(o.b); // 2
// Is there a 'b' own property on o? Yes, and its value is 2.
// The prototype also has a 'b' property, but it's not visited.
// This is called Property Shadowing

console.log(o.c); // 4
// Is there a 'c' own property on o? No, check its prototype.
// Is there a 'c' own property on o.[[Prototype]]? Yes, its value is 4.
```


index.js saving...

```javascript
var o = {
  a: 2,
  m: function(){
    return this.a + 1;
  }
};

console.log(o.m()); // 3
// 当调用 o.m 时，'this' 指向了 o.

var p = Object.create(o);
// p是一个继承自 o 的对象

p.a = 4; // 创建 p 的自身属性 'a'
console.log(o.m());
console.log(p.m());
```



```
var o = {a: 1};

// o 这个对象继承了 Object.prototype 上面的所有属性
// o 自身没有名为 hasOwnProperty 的属性
// hasOwnProperty 是 Object.prototype 的属性
// 因此 o 继承了 Object.prototype 的 hasOwnProperty
// Object.prototype 的原型为 null
// 原型链如下:
// o ---> Object.prototype ---> null

var a = ["yo", "whadup", "?"];

// 数组都继承于 Array.prototype
// (Array.prototype 中包含 indexOf, forEach 等方法)
// 原型链如下:
// a ---> Array.prototype ---> Object.prototype ---> null

function f(){
  return 2;
}

// 函数都继承于 Function.prototype
// (Function.prototype 中包含 call, bind等方法)
// 原型链如下:
```

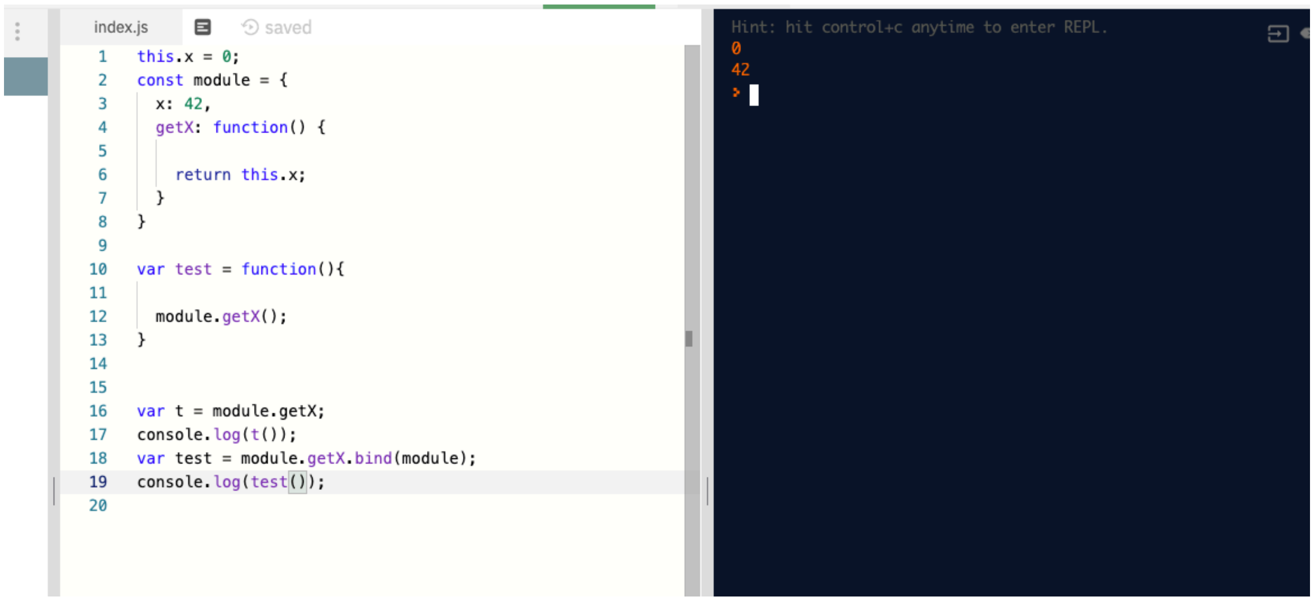


使用 `Object.create` 创建的对象

ECMAScript 5 中引入了一个新方法：`Object.create()`。可以调用这个方法来创建一个新对象。新对象的原型就是调用 create 方法时传入的第一个参数：

```
var a = {a: 1};
// a ---> Object.prototype ---> null

var b = Object.create(a);
// b ---> a ---> Object.prototype ---> null
console.log(b.a); // 1 (继承而来)

var c = Object.create(b);
// c ---> b ---> a ---> Object.prototype ---> null

var d = Object.create(null);
// d ---> null
console.log(d.hasOwnProperty); // undefined, 因为d没有继承Object.prototype
```

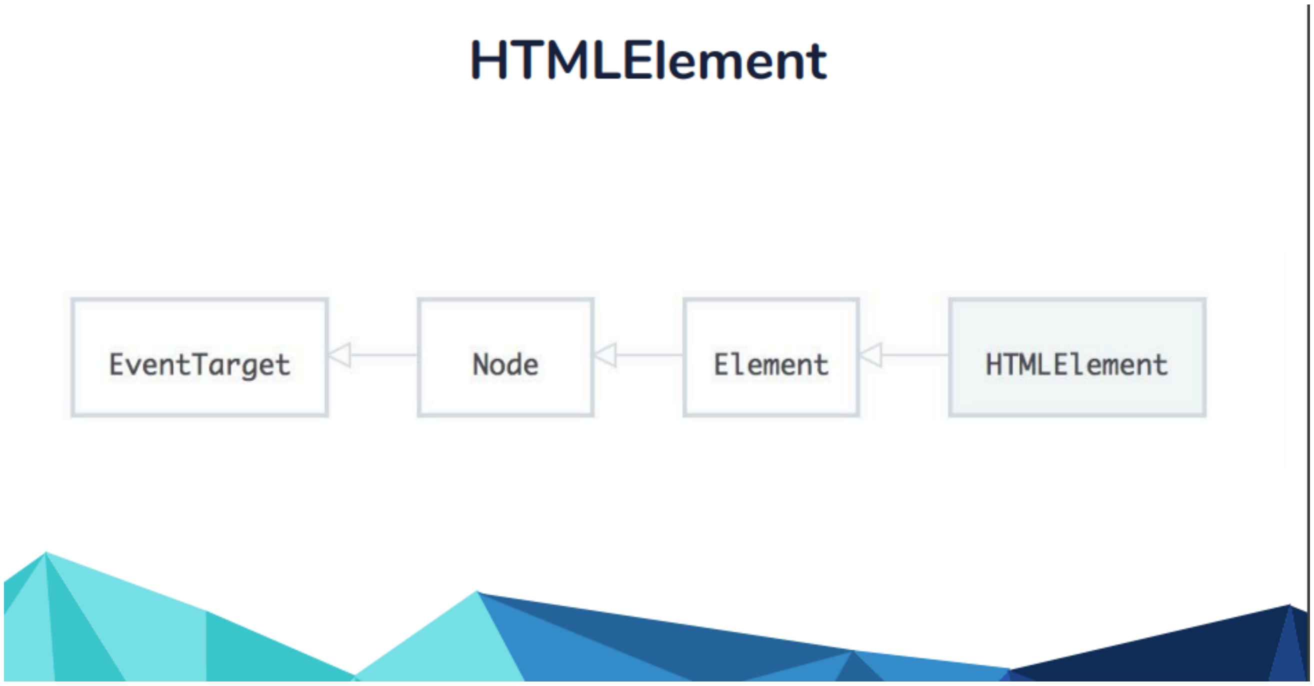
bind 强行绑定this为括号内对象



```
this.x = 0;
const module = {
  x: 42,
  getX: function() {

    return this.x;
  }
}

var test = function(){

  module.getX();
}


var t = module.getX;
console.log(t());
var test = module.getX.bind(module);
console.log(test());
```

```
Hint: hit control+c anytime to enter REPL.
0
42
>
```

DOM



fetch api

promise

[***https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide/Using_promises***](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide/Using_promises)

arguments

`arguments` 对象是所有（非箭头）函数中都可用的局部变量。你可以使用 `arguments` 对象在函数中引用函数的参数。此对象包含传递给函数的每个参数，第一个参数在索引0处。例如，如果一个函数传递了三个参数，你可以以如下方式引用他们:

类似array但是不是array

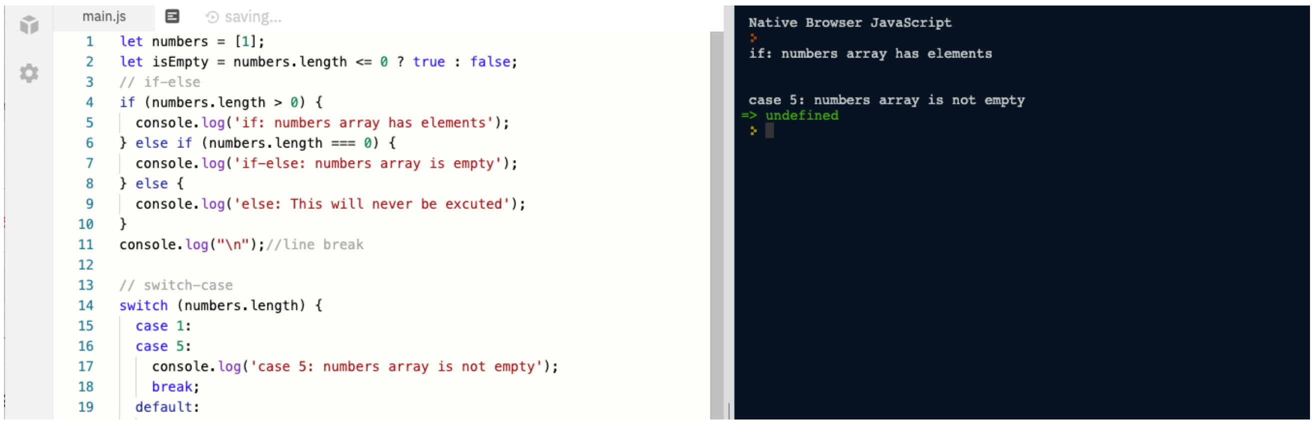
typeof 返回 object

rest operator: ...

functions

```
//Using variable

let addVariable = function (x, y) {

 return x + y;

};

console.log("addVariable: " + addVariable(1, 2));

console.log(""); //line break


//Named functions

function addNamed(x, y) {

 //statements

 return x + y;

}

console.log("addNamed: " + addNamed(5, 5));

console.log(""); //line break


//Anonymous Functions

(function (x, y) {

 return x + y;

})(1, 2); //

// addNamed(1, 2)

//Call a function using ()

function addAndExcute(x, y, callback) { //closure

 let sum = x + y;

 console.log("Sum = " + sum);
```

```
  console.log("typeof callback: " + typeof callback);
  if (typeof callback === 'function') {
   console.log("Excuting callback");
   callback(sum);
  } else {
   console.log("No callback");
  }
}
let fn = function(x) {
  console.log("Callback sum: " + x);
};
addAndExcute(2, 3, fn);
```



main.js   saving...

```
let numbers = [1];
let isEmpty = numbers.length <= 0 ? true : false;
// if-else
if (numbers.length > 0) {
  console.log('if: numbers array has elements');
} else if (numbers.length === 0) {
  console.log('if-else: numbers array is empty');
} else {
  console.log('else: This will never be excuted');
}
console.log("\n");//line break

// switch-case
switch (numbers.length) {
  case 1:
  case 5:
    console.log('case 5: numbers array is not empty');
    break;
  default:
```

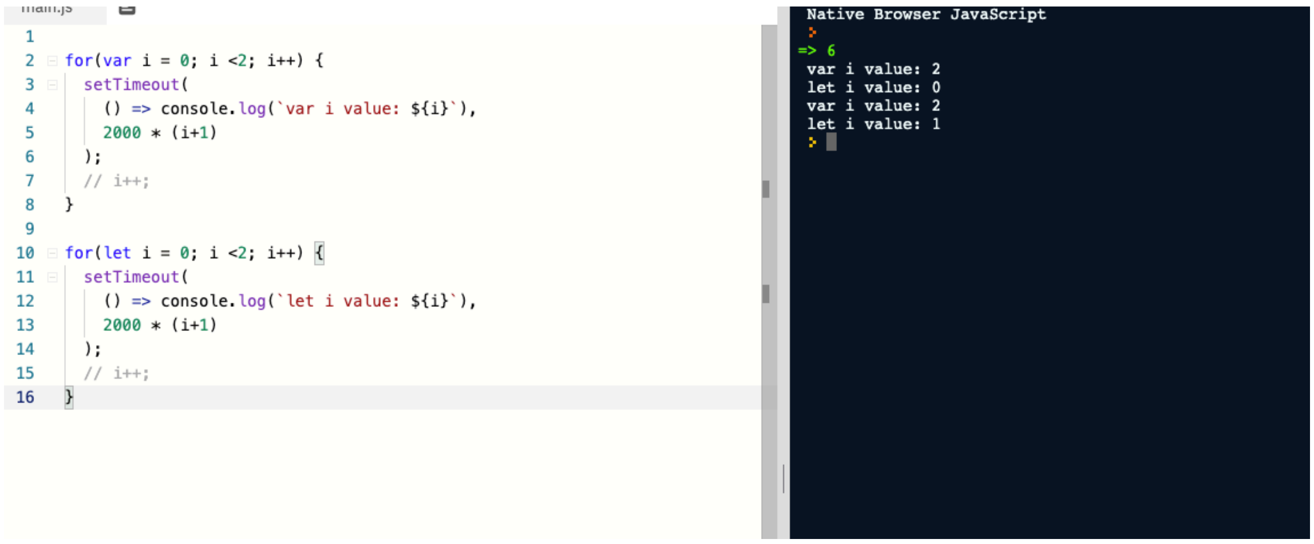
```
Native Browser JavaScript
>
if: numbers array has elements


case 5: numbers array is not empty
=> undefined
>
```

for each

`**slice()**` 方法返回一个新的数组对象，这一对象是一个由 `begin` 和 `end` 决定的原数组的浅拷贝（包括 `begin`，不包括`end`）。原始数组不会被改变。

loop

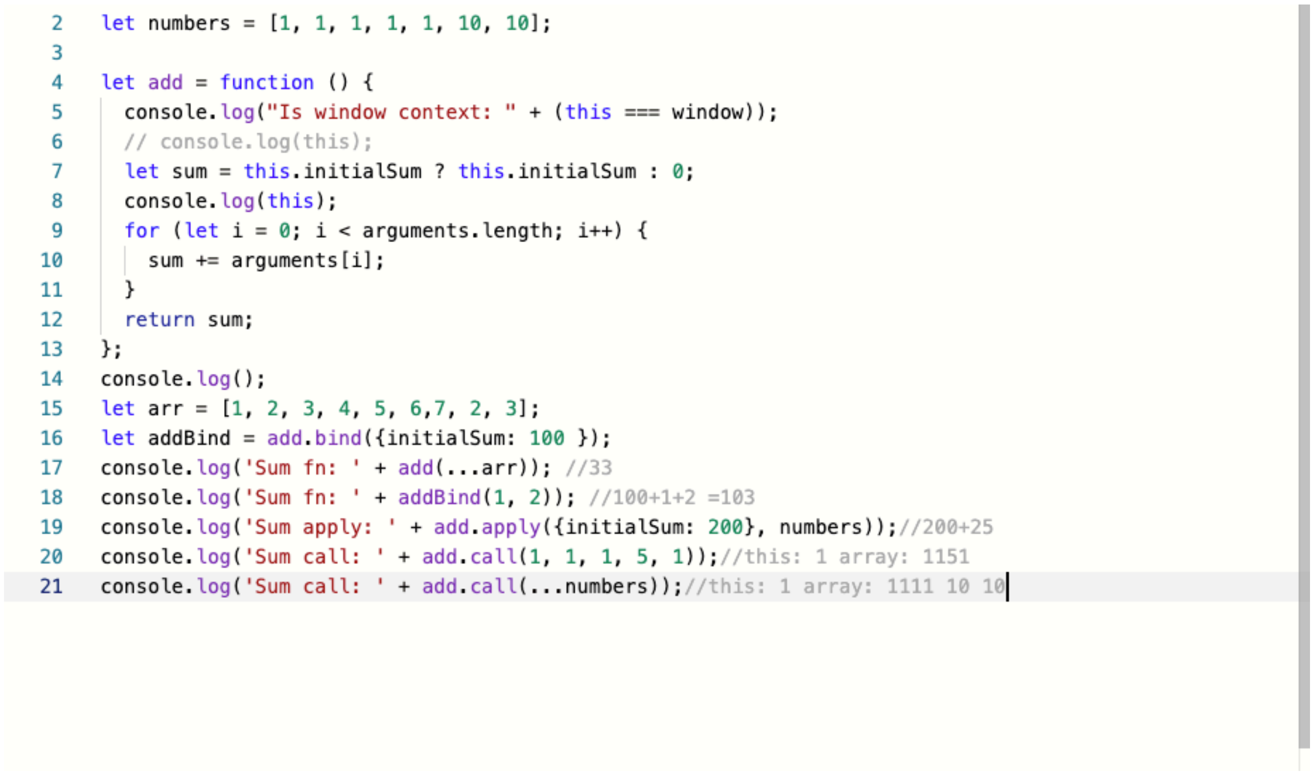
settimeout看成异步方法，先等在执行语句，在等待过程中，全局的i已经变成2。 局部的则是使用传递进去的局部的i



```
for(var i = 0; i <2; i++) {
  setTimeout(
    () => console.log(`var i value: ${i}`),
    2000 * (i+1)
  );
  // i++;
}

for(let i = 0; i <2; i++) {
  setTimeout(
    () => console.log(`let i value: ${i}`),
    2000 * (i+1)
  );
  // i++;
}
```

```
Native Browser JavaScript
=> 6
 var i value: 2
 let i value: 0
 var i value: 2
 let i value: 1
```

dynamic function



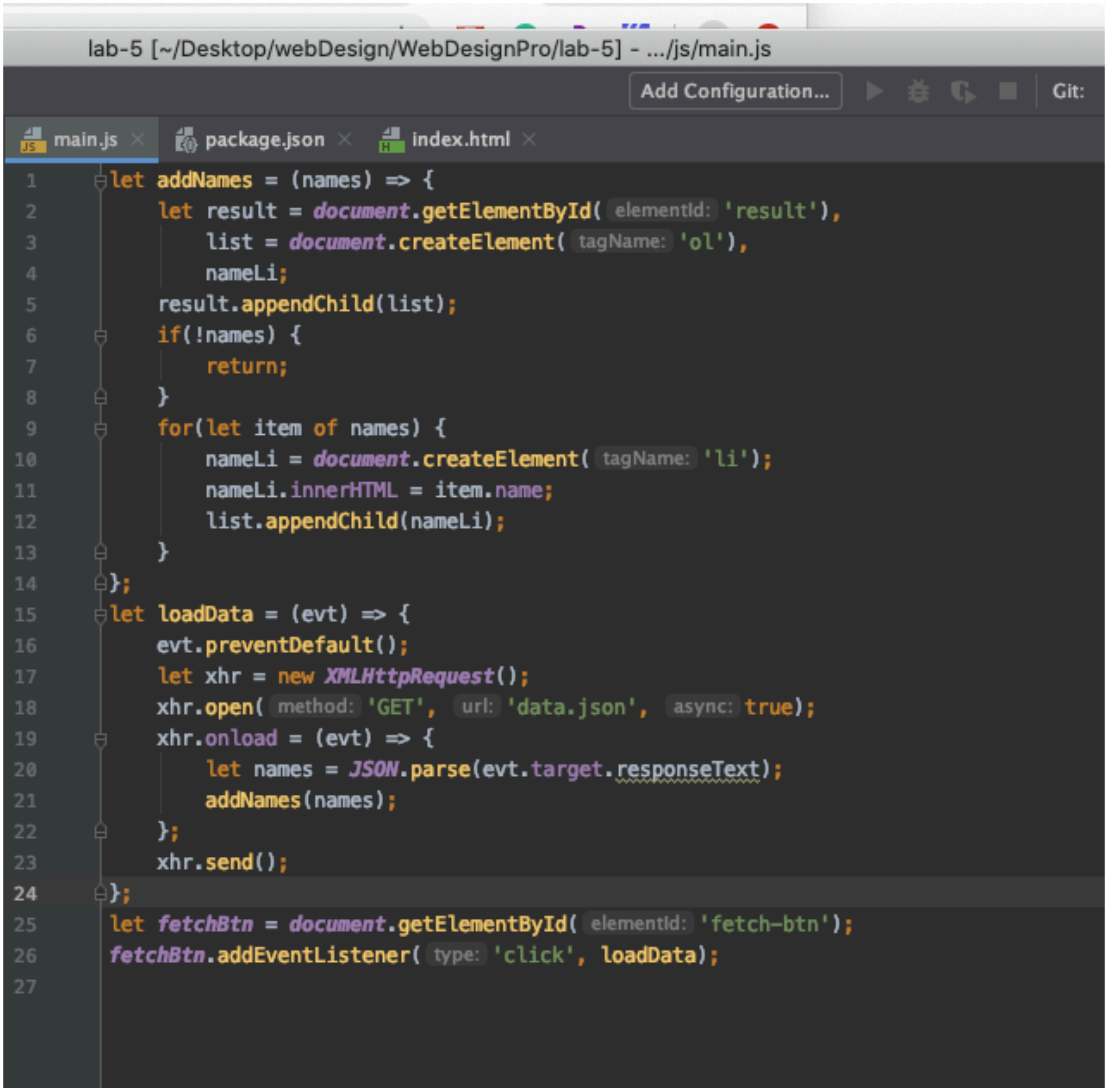
```
let numbers = [1, 1, 1, 1, 1, 10, 10];

let add = function () {
  console.log("Is window context: " + (this === window));
  // console.log(this);
  let sum = this.initialSum ? this.initialSum : 0;
  console.log(this);
  for (let i = 0; i < arguments.length; i++) {
    sum += arguments[i];
  }
  return sum;
};
console.log();
let arr = [1, 2, 3, 4, 5, 6,7, 2, 3];
let addBind = add.bind({initialSum: 100 });
console.log('Sum fn: ' + add(...arr)); //33
console.log('Sum fn: ' + addBind(1, 2)); //100+1+2 =103
console.log('Sum apply: ' + add.apply({initialSum: 200}, numbers));//200+25
console.log('Sum call: ' + add.call(1, 1, 1, 5, 1));//this: 1 array: 1151
console.log('Sum call: ' + add.call(...numbers));//this: 1 array: 1111 10 10
```

call

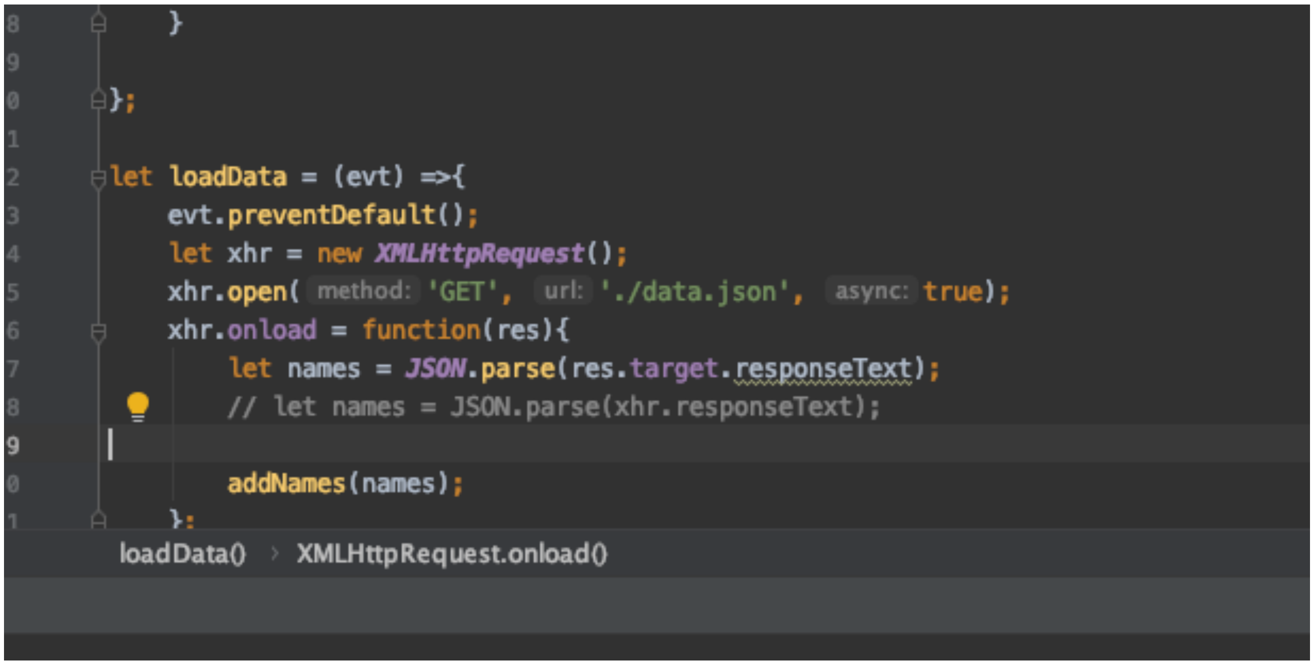
call接收的第一个参数是this， 之后是传递的参数

apply直接就是传递的参数

json相关


lab-5 [~/Desktop/webDesign/WebDesignPro/lab-5] - .../js/main.js
Add Configuration...
Git:
main.js
package.json
index.html

```
let addNames = (names) => {
    let result = document.getElementById( elementId: 'result'),
        list = document.createElement( tagName: 'ol'),
        nameLi;
    result.appendChild(list);
    if(!names) {
        return;
    }
    for(let item of names) {
        nameLi = document.createElement( tagName: 'li');
        nameLi.innerHTML = item.name;
        list.appendChild(nameLi);
    }
};
let loadData = (evt) => {
    evt.preventDefault();
    let xhr = new XMLHttpRequest();
    xhr.open( method: 'GET',  url: 'data.json',  async: true);
    xhr.onload = (evt) => {
        let names = JSON.parse(evt.target.responseText);
        addNames(names);
    };
    xhr.send();
};
let fetchBtn = document.getElementById( elementId: 'fetch-btn');
fetchBtn.addEventListener( type: 'click', loadData);
```



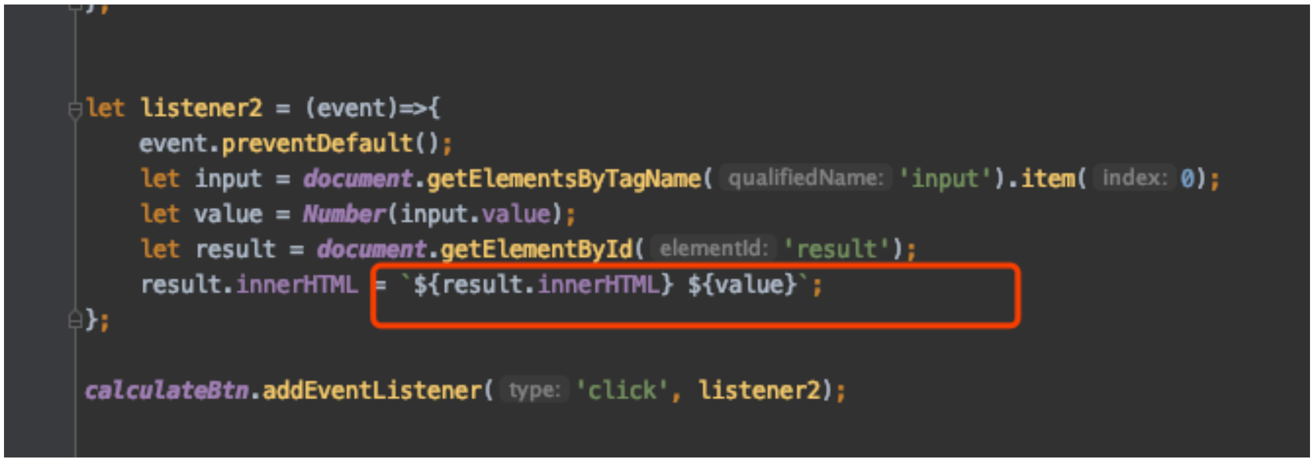
```
    }

};

let loadData = (evt) =>{
    evt.preventDefault();
    let xhr = new XMLHttpRequest();
    xhr.open( method: 'GET', url: './data.json', async: true);
    xhr.onload = function(res){
        let names = JSON.parse(res.target.responseText);
        // let names = JSON.parse(xhr.responseText);

        addNames(names);
    };
```

loadData() › XMLHttpRequest.onload()

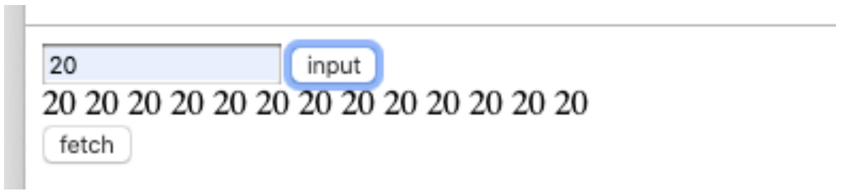
Document.getElementById('')

Document.creatElement(tagname);

Nameli.innerHtml = item.name 给html页面设置值



```
let listener2 = (event)=>{
    event.preventDefault();
    let input = document.getElementsByTagName( qualifiedName: 'input').item( index: 0);
    let value = Number(input.value);
    let result = document.getElementById( elementId: 'result');
    result.innerHTML = `${result.innerHTML} ${value}`;
};

calculateBtn.addEventListener( type: 'click', listener2);
```

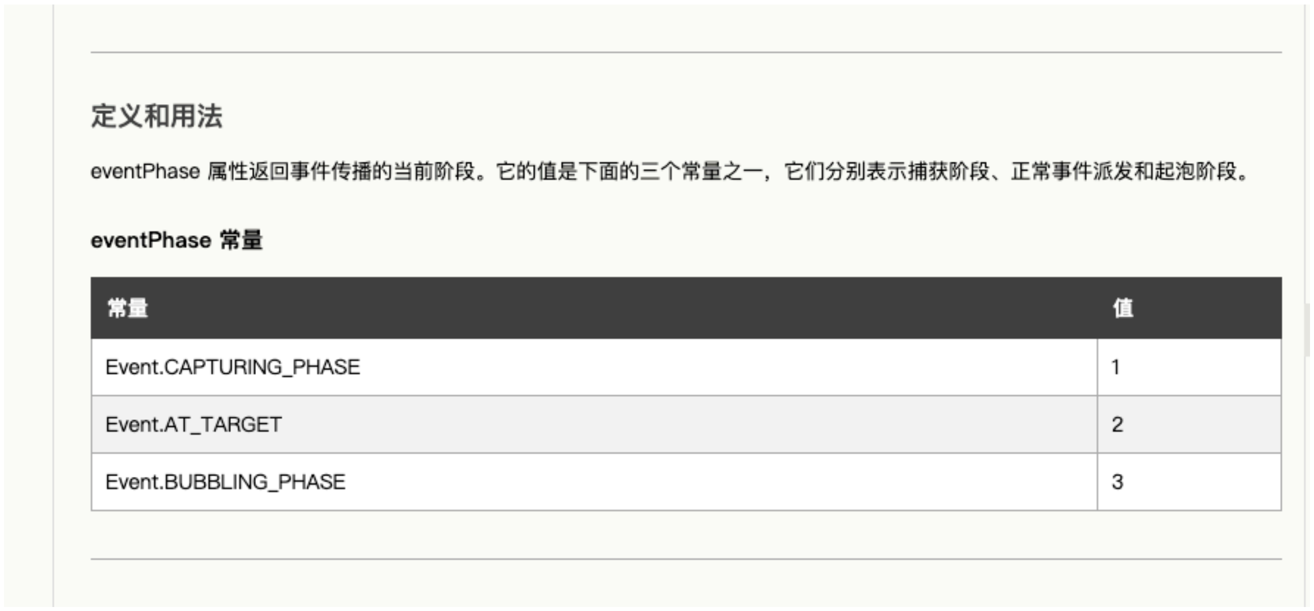


event

https://codepen.io/amuthanarulraj/pen/OOboVX?editors=1111

true： 在捕获阶段触发

否则在冒泡阶段触发


定义和用法

eventPhase 属性返回事件传播的当前阶段。它的值是下面的三个常量之一，它们分别表示捕获阶段、正常事件派发和起泡阶段。

eventPhase 常量

常量	值
Event.CAPTURING_PHASE	1
Event.AT_TARGET	2
Event.BUBBLING_PHASE	3

REGEX

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Character_Classes

\d 阿拉伯数字

\D 非阿拉伯数字

\w 字母和字符

\s 空格

\S 空格以外的单个字符

\t 水平tab

\r 回车

\n linefeed换行符

\v vertical tab

\f 换页

[\b] 退格键， backspace

\0 null character

\cX Matches a control character using [caret notation](), where "X" is a letter from A–Z (corresponding to codepoints `U+0001`–`U+001F`). For example, `/\cM/` matches "\r" in "\r\n".

\xhh   Matches the character with the code hh (two hexadecimal digits).

\ Indicates that the following character should be treated specially, or "escaped". It behaves one of two ways.

assertions https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Assertions

^ begining of input

$ end of input

\b a word boundary. This is the position where a word character is not followed or preceded by another word-character, such as between a letter and a space.

\B matchs a non-word boundary.

x(?=y) Matches "x" only if "x" is followed by "y" xy

x(?!y) Matches "x" only if "x" is not followed by "y".

(?<=y)x Matches "x" only if "x" is preceded by "y".   Yx

(?<!y)x Matches "x" only if "x" is not preceded by "y".

groups and ranges

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Groups_and_Ranges

(x) Matches `*x*` and remembers the match. For example, `/(foo)/` matches and remembers "foo" in "foo bar".

\n Where "n" is a positive integer. A back reference to the last substring matching the n parenthetical in the regular expression (counting left parentheses). For example, `/apple(,)\sorange\1/` matches "apple, orange," in "apple, orange, cherry, pea

`(?x)`	*Named capturing group: Matches "x" and stores it on the groups property of the returned matches under the name specified by* `` `. The angle brackets ( `` < `` and `` > `` ) are required for group name.For example, to extract the United States area code from a phone number, we could use `` */((?\d\d\d))/* `` . The resulting number would appear under `` *matches.groups.area`.*

(?:x)   Non-capturing group: Matches "x" but does not remember the match. The matched substring cannot be recalled from the resulting array's elements ([1], ..., [n]) or from the predefined RegExp object's properties ($1, ..., $9).
Examples

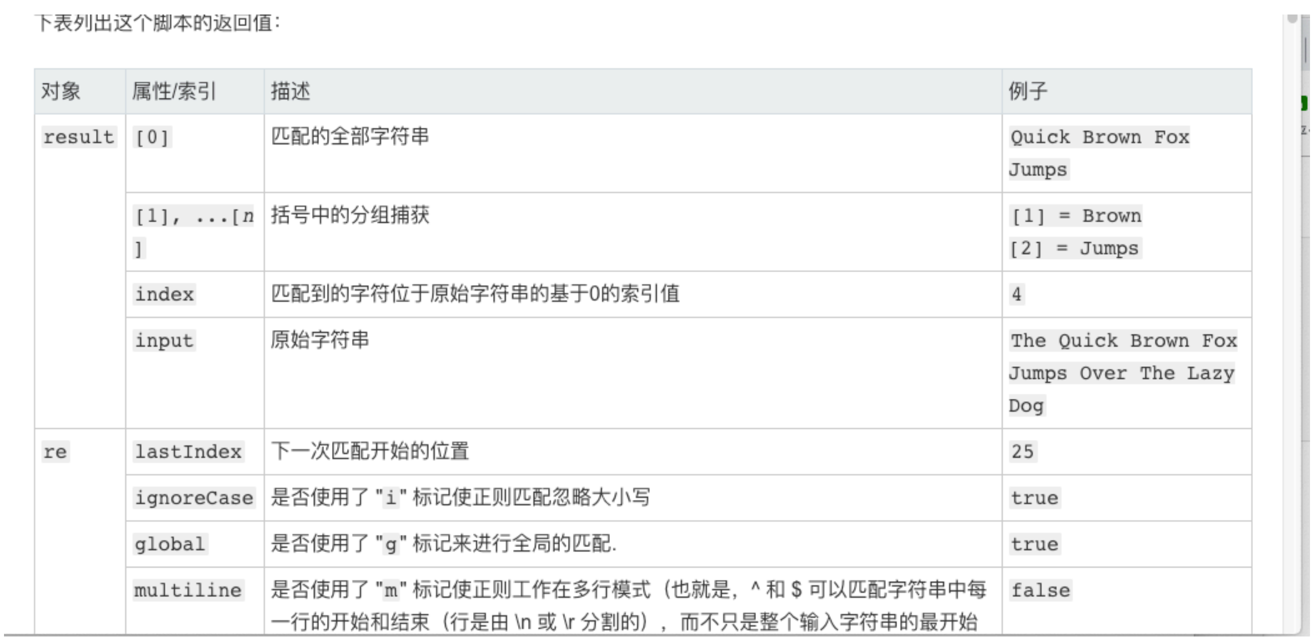
quantifiers

*x**	*Matches the preceding item "x" 0 or more times. For example, `/bo*/` matches "boooo" in "A ghost booooed" and "b" in "A bird warbled", but nothing in "A goat grunted".*
`x+`	*Matches the preceding item "x" 1 or more times. Equivalent to `{1,}`. For example, `/a+/` matches the "a" in "candy" and all the "a"'s in "caaaaaaandy".*
`x?`	*Matches the preceding item "x" 0 or 1 times. For example, `/e?le?/` matches the "el" in "angel" and the "le" in "angle."If used immediately after any of the quantifiers `*`, `+`, `?`, or `{}`, makes the quantifier non-greedy (matching the minimum number of times), as opposed to the default, which is greedy (matching the maximum number of times).*
`x{n}`	*Where "n" is a positive integer, matches exactly "n" occurrences of the preceding item "x". For example, `/a{2}/` doesn't match the "a" in "candy", but it matches all of the "a"'s in "caandy", and the first two "a"'s in "caaandy".*
`x{n,}`	*Where "n" is a positive integer, matches at least "n" occurrences of the preceding item "x". For example, `/a{2,}/` doesn't match the "a" in "candy", but matches all of the a's in "caandy" and in "caaaaaaandy".*
`x{n,m}`	*Where "n" is 0 or a positive integer, "m" is a positive integer, and `*m* > *n*`, matches at least "n" and at most "m" occurrences of the preceding item "x". For example, `/a{1,3}/` matches nothing in "cndy", the "a" in "candy", the two "a"'s in "caandy", and the first three "a"'s in "caaaaaaandy". Notice that when matching "caaaaaaandy", the match is "aaa", even though the original string had more "a"s in it.*

`x*?`
`x+?`
`x??`
`x{n}?`
`x{n,}?`
`x{n,m}?`
	By default quantifiers like `*` and `+` are "greedy", meaning that they try to match as much of the string as possible. The `?` character after the quantifier makes the quantifier "non-greedy": meaning that it will stop as soon as it finds a match. For example, given a string like "some new thing": `/<.*>/` will match " new " `/<.*?>/` will match ""

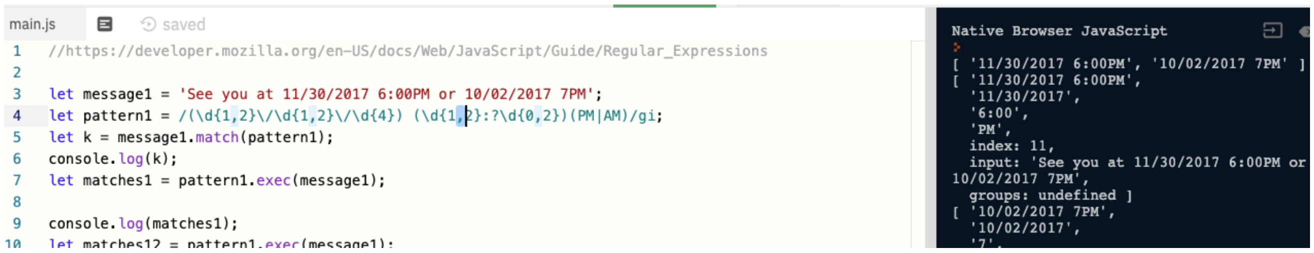
By default quantifiers like `*` and `+` are "greedy", meaning that they try to match as much of the string as possible. The `?` character after the quantifier makes the quantifier "non-greedy

exec的返回值



下表列出这个脚本的返回值：

对象	属性/索引	描述	例子
`result`	`[0]`	匹配的全部字符串	`Quick Brown Fox Jumps`
	`[1], ...[n ]`	括号中的分组捕获	`[1] = Brown` `[2] = Jumps`
	`index`	匹配到的字符位于原始字符串的基于0的索引值	`4`
	`input`	原始字符串	`The Quick Brown Fox Jumps Over The Lazy Dog`
`re`	`lastIndex`	下一次匹配开始的位置	`25`
	`ignoreCase`	是否使用了 "`i`" 标记使正则匹配忽略大小写	`true`
	`global`	是否使用了 "`g`" 标记来进行全局的匹配.	`true`
	`multiline`	是否使用了 "`m`" 标记使正则工作在多行模式（也就是，^ 和 $ 可以匹配字符串中每一行的开始和结束（行是由 \n 或 \r 分割的），而不只是整个输入字符串的最开始	`false`



main.js saved

```javascript
//https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

let message1 = 'See you at 11/30/2017 6:00PM or 10/02/2017 7PM';
let pattern1 = /(\d{1,2}\/\d{1,2}\/\d{4}) (\d{1,2}:?\d{0,2})(PM|AM)/gi;
let k = message1.match(pattern1);
console.log(k);
let matches1 = pattern1.exec(message1);

console.log(matches1);
let matches12 = pattern1.exec(message1);
```

Native Browser JavaScript

```
[ '11/30/2017 6:00PM', '10/02/2017 7PM' ]
[ '11/30/2017 6:00PM',
  '11/30/2017',
  '6:00',
  'PM',
  index: 11,
  input: 'See you at 11/30/2017 6:00PM or
10/02/2017 7PM',
  groups: undefined ]
[ '10/02/2017 7PM',
  '10/02/2017',
  '7',
```